Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Коваленка Олександра Сергійовича*
(ПІБ)

академічної групи *121-19ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення моделювання оптимального раціону харчування на C#*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	*проф. Бердник М.Г.*			
розділів:				
спеціальний	*проф. Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц.Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19ск-1* *Коваленка Олександра Сергійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення моделювання оптимального раціону харчування на C#*

затверджена наказом ректора НТУ «ДП» від «18» травня 2022 р. № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *проф. Бердник М.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Коваленко О.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 89 с., 42 рис., 13 табл., 3 дод., 22 джерела.

Об'єкт розробки: програмний додаток для моделювання оптимального раціону харчування на C#.

Мета кваліфікаційної роботи: розробка програмного додатку для моделювання оптимального раціону харчування, який дозволить ефективно контролювати рівень споживання вуглеводів, що, в свою чергу, дозволить контролювати рівень глюкози в крові та попередити виникнення різноманітних захворювань, наприклад – цукрового діабету та супутніх до нього хвороб.

У вступі наводиться аналіз сучасного стану проблеми, проводиться уточнення постановки завдання, мети кваліфікаційної роботи та галузі її застосування, а також обґрунтовується актуальність теми.

У першому розділі виконується дослідження предметної галузі та наявних рішень заданої задачі. Визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі проводиться вибір платформи для розробки, виконується проектування та розробка програмного додатку. Наводиться опис алгоритму та архітектури системи, характеристик складу технічних засобів, визначаються вхідні і вихідні дані, описується процес роботи програми.

В економічному розділі обчислюється трудомісткість розробленого програмного продукту, виконується обчислення вартості роботи зі створення програмного додатку та визначається час на його створення.

Практичне значення роботи полягає у наданні людині можливості контролювати та планувати обсяги споживання вуглеводів шляхом створення оптимального раціону харчування базуючись на загальній максимальній вартості продуктового кошику, кількості вуглеводів в продуктах харчування та тривалості часового проміжку, на яких планується створення оптимального раціону.

Актуальність розробки даного програмного продукту обумовлюється відсутністю на ринку аналогічних засобів для оптимізації раціону харчування та високим рівнем складності ручних обчислень подібного роду.

Список ключових слів: РАЦІОН ХАРЧУВАННЯ, ХЛІБНА ОДИНИЦЯ, ТОВАР, ОПТИМІЗАЦІЯ, ПРОГРАМНИЙ ДОДАТОК, C#.

ABSTRACT

Explanatory note: 89 pp., 42 fig., 13 table, 3 appendix, 22 sources.

Object of development: software application for modeling the optimal diet in C #.

The purpose of the qualification work: to develop a software application for modeling the optimal diet, which will effectively control the level of carbohydrate intake, which, in turn, will control blood glucose levels and prevent various diseases such as diabetes and related diseases.

The introduction provides an analysis of the current state of the problem, clarifies the task, the purpose of the qualification work and the field of its application, as well as substantiates the relevance of the topic.

The first section examines the subject area and the available solutions to the problem. The urgency of the task and the purpose of development is determined, the task statement is developed.

In the first section the research of the subject area and available solutions of the given problem is carried out. The urgency of the task and the purpose of development is determined, the task statement is developed.

The second section selects the platform for development, performs design and development of software applications. The description of algorithm and architecture of system, characteristics of structure of technical means is given, input and output data are defined, the process of work of the program is described.

The economic section calculates the complexity of the developed software product, calculates the cost of creating a software application and determines the time for its creation.

The practical value of the work is to enable people to control and plan carbohydrate intake by creating an optimal diet based on the total maximum value of the food basket, the amount of carbohydrates in food and the length of time for which you plan to create an optimal diet.

The relevance of the development of this software product is due to the lack of similar tools on the market to optimize the diet and the high level of complexity of manual calculations of this kind.

List of keywords: DIET, BREAD UNIT, GOODS, OPTIMIZATION, SOFTWARE APPLICATION, C #.

ЗМІСТ

РЕФЕРАТ 3
ABSTRACT 4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ 10
1.1. Загальні відомості з предметної галузі 10
1.2. Призначення розробки та галузь застосування 11
1.3. Підстава для розробки 12
1.4. Постановка завдання 12
1.5. Вимоги до функціональних характеристик 14
1.6. Вимоги до інформаційної безпеки 14
1.7. Вимоги до складу та параметрів технічних засобів 15
1.8. Вимоги до інформаційної та програмної сумісності 15
РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ . 16
2.1. Функціональне призначення програми 16
2.2. Опис застосованих математичних методів 18
2.2.1. Математична модель 18
2.2.2. Метод оптимізації 19
2.3. Опис використаної архітектури та шаблонів проектування 20
2.4. Опис використаних технологій та мов програмування 23
2.5. Опис структури програми та алгоритмів її функціонування 28
2.6. Обґрунтування та організація вхідних та вихідних даних програми 37
2.7. Опис розробленого програмного продукту 38
2.7.1. Використані технічні засоби 38

2.7.2. Використані програмні засоби 38
2.7.3. Виклик та завантаження програми 39
2.7.4. Опис інтерфейсу користувача 39
РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ 59
3.1. Обчислення трудомісткості розробки програмного забезпечення 59
3.2. Обчислення витрат на створення програмного забезпечення 62
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 67
ДОДАТОК А 70
ДОДАТОК Б 88
ДОДАТОК В 89

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ХО – хлібна одиниця;

ІМТ – індекс маси тіла;

WPF - Windows Presentation Foundation;

MVVM – Model-View-ViewModel;

ООП – об’єктно-орієнтоване програмування;

EF – Entity Framework;

LINQ – Language Integrated Query.

ВСТУП

Цукровий діабет – це захворювання, яке зумовлюються підвищенням рівня глюкози в крові через частковий або повний брак гормону інсуліну. Цукровий діабет може спричиняти низку інших захворювань, ураження внутрішніх органів, судин, нервової системи тощо.

Розрізняють інсулінозалежний (перший) та інсулінонезалежний (другий) типи цукрового діабету. Цукровий діабет другого типу вважається більш легкою формою хвороби та зазвичай спричинений неправильним харчуванням, в особливості наявністю великої кількості продуктів, багатих на жири та солодощів.

Зважаючи на причини виникнення захворювання для хворих на цукровий діабет другого типу вкрай важливо мати збалансований раціон харчування, зокрема, не перевищувати допустимі норми щоденного вживання вуглеводів, адже вони найсильніше впливають на підвищення рівня цукру в крові.

Для полегшення обліку кількості вуглеводів для споживання хворими на цукровий діабет у 1990-х роках було створено термін «хлібна одиниця».

Хлібна одиниця (або вуглеводна одиниця) – це штучний параметр для оцінки вмісту вуглеводів в конкретному продукті харчування. Одна хлібна одиниця рівнозначна 10 г простих вуглеводів в продукті або 12 г вуглеводів включаючи клітковину.

Контроль та спостереження за споживанням вуглеводів (обчислення кількості грамів вуглеводів в їжі або хлібних одиниць) є головною стратегією для досягнення цільового рівня глюкози в крові.

Задача оптимізації харчування хворих на цукровий діабет людей є досить трудомісткою та потребує значних інженерних обчислень із врахуванням багатьох додаткових факторів – вік, стать, вага, кількість хлібних одиниць в продуктах харчування тощо. Розрахувати оптимальний склад продуктового кошику самостійно для багатьох людей, яким необхідно власноруч контролювати рівень цукру в крові, майже неможливо. Це робить актуальною

необхідність розробки програмного засобу, який максимально спростить та пришвидшить процес таких обчислень.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Хлібна одиниця (або вуглеводна одиниця) – це штучний параметр для оцінки вмісту вуглеводів в конкретному продукті харчування [1]. Цей термін було створено німецькими дієтологами для оцінки раціону діабетиків в 1990-х роках. Одна хлібна одиниця рівнозначна 10 г простих вуглеводів в продукті або 12 г вуглеводів включаючи клітковину.

Щоденне обмеження щодо споживання вуглеводів при цукровому діабеті будь-якого типу становить 18-24 ХО [2].

У випадку, коли індекс маси тіла (ІМТ) людини не перевищує показник у 18,49 одиниць – тобто спостерігається недостатність ваги – щоденне обмеження споживання вуглеводів становить 18-30 ХО. Якщо ІМТ перевищує 35 одиниць – кількість вуглеводів на день становить 10 ХО. При ІМТ вище 40 одиниць – 6-8 ХО на день.

Окрім цих рекомендацій існують обмеження споживання вуглеводів в залежності від віку та статі людини. Рекомендовані лікарями обмеження щодо споживання вуглеводів наведено в таблиці 1.1 [3].

Таблиця 1.1

Щоденне обмеження споживання вуглеводів у хлібних одиницях в залежності від віку та статі

Вік, років	Кількість ХО на день
4-6	12-13
7-10	15-16
11-14	18-20 (хлопчики)
11-14	16-17 (дівчата)
15-18	19-21 (хлопчики)
15-18	17-18 (дівчата)

Закін. табл. 1.1

Вік, років	Кількість ХО на день
Дорослі	20-22

Також є рекомендації щодо споживання вуглеводів в залежності від статі людини та рівня її фізичного навантаження. Рекомендовані лікарями обмеження наведені в таблиці 1.2 [4].

Таблиця 1.2

Щоденне обмеження споживання вуглеводів у хлібних одиницях в залежності від статі та фізичного навантаження

Фізичне навантаження	Стать	Середня добова норма вуглеводів, ХО
Важке	Чоловіки та жінки	23-28
Середнє	Чоловіки	19-22
	Жінки	16-19
Низьке	Чоловіки	14-16
	Жінки	10-14

1.2. Призначення розробки та галузь застосування

На відміну від наявних у вільному доступі різноманітних засобів обчислення кількості хлібних одиниць засобів, створюваний програмний додаток надає можливість оптимізації продуктового кошику хворого на цукровий діабет другого типу цілком, а не окремих продуктів. При оптимізації враховуються кількість хлібних одиниць в продукті, максимальна вартість продуктового кошику та щоденні обмеження на споживання вуглеводів.

Програма розрахована на побутове використання людьми хворими на цукровий діабет другого типу. Це надає можливість зменшити навантаження на травну систему людини та може знизити ризики виникнення інших, супутніх хвороб, пов’язаних з неправильним харчуванням.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи) є:

– освітня програма 121 Інженерія програмного забезпечення;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;

– завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення моделювання оптимального раціону харчування на C#».

1.4. Постановка завдання

Розробити програмний додаток мовою програмування C# для автоматизації процесу моделювання оптимального раціону харчування при цукровому діабеті 2 типу.

Функціональні можливості

Усі необхідні дані централізовано зберігаються у базі даних програмного додатку.

При роботі з програмним додатком користувач може:

- вводити дані для розрахунку;
- переглядати список продуктів для розрахунків;
- маніпулювати списком продуктів для розрахунків: редагування (вартість), додавання, видалення;
- переглядати список обмежень щодо споживання продуктів;
- додавати обмеження щодо споживання продуктів: більше ніж, менше ніж, рівно певній кількості або масі;
- переглядати результати розрахунку продуктового кошику;
- створювати власний кабінет;
- редагувати персональні дані у власному кабінеті;
- зберігати персоналізований список продуктів та обмежень щодо їхнього споживання;

– зберігати кілька списків продуктів.

При роботі з програмним додатком адміністратор має можливість:

– переглядати список продуктів для продуктового кошику та їх характеристик за замовчуванням;

– редагувати список продуктів для продуктового кошику та їх характеристик за замовчуванням;

– переглядати список користувачів;

– видаляти користувачів.

Передбачається, що програмна система налічуватиме наступні об’єкти з відповідними властивостями:

а) Користувач:

1) стать;
2) вага;
3) зріст;
4) ступінь фізичного навантаження (низька, середня, важка).

б) Продукт:

1) назва;
2) категорія;
3) вартість;
4) одиниця виміру;
5) кількість хлібних одиниць на одиницю виміру;
6) стан продукту (сирий, приготовлений певним чином тощо);
7) обмеження щодо споживання.

в) Продуктовий кошик:

1) дані користувача;
2) список продуктів;
3) термін, на який було проведено розрахунок математичної моделі (день, тиждень, місяць тощо);
4) максимальна вартість продуктового кошику.

1.5. Вимоги до функціональних характеристик

Програмний додаток повинен бути реалізований мовою програмування C# у вигляді клієнтського додатку для операційної системи Windows, та мати графічний користувальницький інтерфейс. Передбачається використання технологій Windows Forms або WPF та .NET Framework.

Інформація, якою оперує програмний додаток повинна зберігатися в базі даних на сервері.

Програма повинна надавати можливість з оптимізації змісту продуктового кошику користувача виходячи із заданої максимальної вартості кошику та денних потреб людини у споживанні хлібних одиниць при цукровому діабеті другого типу.

1.6. Вимоги до інформаційної безпеки

Програмний додаток повинен задовольняти основним принципам інформаційної безпеки:

- розмежування доступу користувачів до системи;
- забезпечення цілісності даних;
- відповідність принципу «усе, що не дозволено – заборонено»;
- надання користувачам мінімальних привілеїв, достатніх для виконання дій згідно з їхньою роллю в системі.

За замовчуванням будь-який користувач при запуску програмного додатку є гостем та може виконувати обмежену кількість операцій в системі. Для отримання більш широкого спектру можливостей користувачу варто авторизуватися в системі використовуючи свій унікальний логін та пароль. У випадку використання програмного додатку вперше користувач матиме можливість зареєструватися в ньому для створення особистого кабінету.

Поля для введення даних повинні мати відповідні функції для перевірки коректності введених даних та надавати відповідні повідомлення про некоректне введення при спробі введення хибних даних.

1.7. Вимоги до складу та параметрів технічних засобів

Для роботи програмного додатку рекомендується мати персональний комп’ютер або ноутбук з наступними мінімальними параметрами:

- підтримка 64-розрядного процесору та операційної системи;
- операційна система Windows 10;
- оперативний запам’ятовувальний пристрій об’ємом 2 ГБ або більше;
- процесор з тактовою частотою 1 ГГц або вище;
- DirectX 12;
- жорсткий диск об’ємом не менше 20 ГБ;
- наявність маніпулятора-миші та клавіатури;
- .Net Framework версії 3.0 або вище.

1.8. Вимоги до інформаційної та програмної сумісності

Програмний продукт повинен бути розроблений з використанням об’єктно орієнтованої мови програмування C# у вигляді візуального проекту Windows Forms або WPF.

Необхідно забезпечити функціонування розробленого додатку на персональних комп’ютерах або ноутбуках під керуванням операційних систем Windows 7 та Windows 8.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Розроблюваний програмний додаток забезпечує можливість ведення бази даних продуктів харчування та їхніх характеристик для подальшого створення та оптимізації раціону харчування.

Окрім ведення даних про продукти, також надаються наступні функціональні можливості:

– ведення даних про користувачів – облік персональних даних користувача, необхідних для подальших обчислень та оптимізації раціону харчування;

– ведення даних про продуктові кошики – облік даних про кількість продукту харчування, його ціну та вміст вуглеводів для створення оптимального поєднання таких продуктів виходячи з переліку користувальницьких вимог та обмежень;

– ведення даних про користувальницькі обмеження – облік даних про максимальну вартість створюваного продуктового кошику та тривалість часового проміжку, на який створюється раціон харчування;

– додавання нових продуктів харчування із вказанням їхньої назви, вартості, кількості вуглеводів, стану та категорії, а також кількості та одиниць виміру;

– додавання нових обмежень щодо споживання продуктів харчування з визначенням назви обмеження, його типу, кількості товару та одиниць його виміру.

Для користувача, який не має облікового запису в системі передбачаються наступні можливості:

– створення облікового запису, під час якого необхідно буде ввести ім’я користувача в системі, пароль, вік, стать, зріст та вагу;

– авторизація в додатку за допомогою пароля та імені користувача для отримання повного функціоналу програмного додатку;

– перегляд списку стандартних продуктів;

– перегляд списку стандартних обмежень до продуктів;

– оптимізація раціону харчування на основі списку рекомендованих продуктів;

– перегляд результатів оптимізації.

Авторизованому користувачеві доступний такий функціонал:

– перегляд персональних даних в особистому кабінеті;

– вихід з облікового запису;

– перегляд списку стандартних продуктів харчування;

– перегляд списку стандартних обмежень;

– додавання, редагування та видалення користувальницького продукту харчування;

– додавання, редагування та видалення користувальницького обмеження для власного, не стандартного продукту ;

– перегляд списку користувальницьких продуктів харчування;

– перегляд списку користувальницьких обмежень споживання продуктів;

– оптимізація продуктового кошику на основі власних і стандартних продуктів харчування та їхніх обмежень;

– перегляд результатів оптимізації;

– збереження результатів оптимізації;

– перегляд списку оптимізованих продуктових кошиків;

– видалення обраного продуктового кошика зі списку.

Користувач з обліковим записом адміністратора має можливість:

– переглядати персональні дані в особистому кабінеті;

– вийти з облікового запису;

– додавати, редагувати та видаляти стандартні продукти харчування;

– додавати, редагувати та видаляти стандартні обмеження щодо споживання рекомендованих продуктів харчування;

– переглядати список наявних користувачів системи;

– видаляти обліковий запис користувача;

– виконувати оптимізацію раціону харчування на основі рекомендованих продуктів та обмежень;

– переглядати результати оптимізації та зберігати їх;

– переглядати список створених продуктових кошиків;

– перегляд списку стандартних продуктів харчування;

– перегляд списку користувальницьких продуктів харчування;

– перегляд стандартних обмежень щодо споживання продуктів;

– перегляд користувальницьких обмежень щодо споживання продуктів;

– авторизація та реєстрація в програмному додатку для розширення функціоналу.

Розроблений програмний додаток надає користувачеві, який не має облікового запису в програмі, можливість зменшити ризик появи цукрового діабету, або супутніх до нього захворювань травної та інших життєво важливих систем організму шляхом створення оптимального раціону харчування направленого на додержання рекомендованих норм денного споживання вуглеводів виходячи із стандартного набору продуктів та обмежень щодо їхнього споживання.

2.2. Опис застосованих математичних методів

2.2.1. Математична модель

Вивчимо поведінку споживача, який має намір за добу закупити продукти в кількості: x_i, $i = 0{,}1 \dots n-1$, за ціною p_i. На закупку товарів споживач виділяє z гривень. Вказані величини повинні бути зв’язані співвідношенням (2.1):

$$\sum_{i=0}^{n-1} p_i x_i = z, z > 0 \quad (2.1)$$

На суму z товари можуть бути закуплені неоднозначно. Тому з усіх варіантів закупок споживач повинен обрати найкращий.

Для цього покупець повинен обрати стратегію закупок (2.2):

$$z \to min$$
$$\mathrm{P}_1 < \sum_{i=0}^{n-1} x_i * \mathrm{XO}_i < \mathrm{P}_2 \quad (2.2)$$

де P_1 – мінімальна кількість хлібних одиниць,

P_2 – максимальна кількість хлібних одиниць,

XO_i – кількість хлібних одиниць в і-му продукті.

2.2.2. Метод оптимізації

У якості методу оптимізації математичної моделі було обрано послідовну версію методу найменших квадратів з методології квадратичного програмування.

Послідовне квадратичне програмування – один з найбільш ефективних алгоритмів загального призначення, головною ідеєю якого є послідовне вирішення задач квадратичного програмування, апроксимуючих певну задачу оптимізації [5]. Для вирішення задачі с обмеженнями метод послідовного квадратичного програмування перетворюється на спеціальну реалізацію ньютонівських методів вирішення системи Лагранжа.

При задачі лінійного програмування вигляду (2.3)

$$\min_{\mathrm{x}} \mathrm{f(x)}, \quad (2.3)$$

з обмеженнями (2.4) та (2.5)

$$\mathrm{b(x)} \geq 0 \quad (2.4)$$
$$\mathrm{c(x)} = 0 \quad (2.5)$$

функція Лагранжа прийме вигляд (2.6)

$$\mathrm{L(x, \lambda, \sigma)} = \mathrm{f(x)} - \lambda^{\mathrm{T}} \mathrm{b(x)} - \sigma^{\mathrm{T}} \mathrm{c(x)}, \quad (2.6)$$

де λ та σ – множники Лагранжа.

Під час ітерації x_k головного алгоритму визначаються відповідні шляхи пошуку d_k в якості рішення підзадачі квадратичного програмування вигляду (2.7)

$$\min_d L(x_k, \lambda_k, \sigma_k) + \nabla L(x_k, \lambda_k, \sigma_k)^T d + \frac{1}{2} d^T \nabla^2_{xx} L(x_k, \lambda_k, \sigma_k) d, \quad (2.7)$$

з обмеженнями (2.8) та (2.9)

$$b(x_k) + \nabla b(x_k)^T d \geq 0, \quad (2.8)$$
$$c(x_k) + + \nabla c(x_k)^T d = 0 \quad (2.9)$$

Сутність методу найменших квадратів

Метод найменших квадратів призначений для вирішення задачі пошуку значень змінної x таким чином, щоб кожне з них на первній ітерації i=1,…m було максимально близько до певного значення y_i. Сутність методу найменших квадратів описується виразом (2.10)

$$\sum_i e_i^2 = \sum_i (y_i - f_i(x))^2 \to \min_x \quad (2.10)$$

2.3. Опис використаної архітектури та шаблонів проектування

Програмний додаток реалізовано на основі архітектурного шаблону проектування MVVM (Model – View – ViewModel), який передбачає розмеження візуальної частини додатку від внутрішньої логіки його роботи [6]. В цілому, логіка роботи методології MVVM наведена на рис. 2.1.

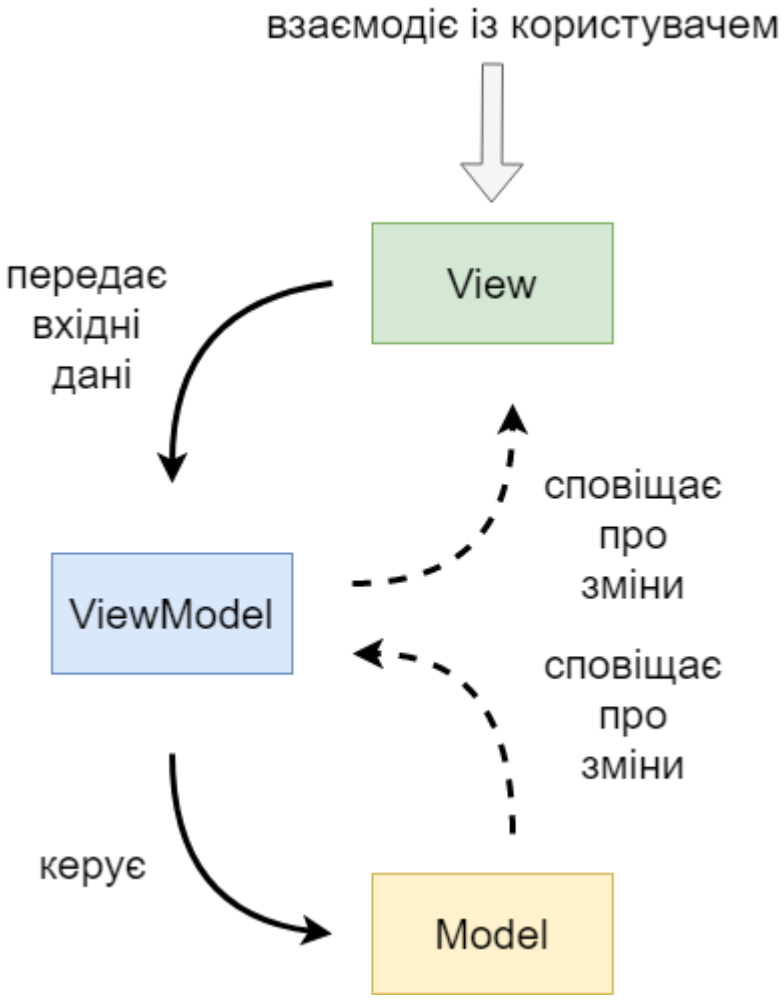


Рис. 2.1. Взаємодія компонентів шаблону MVVM

Як можна побачити на рис. 2.1, кожен компонент моделі відповідає за свою конкретну задачу:

1. Model – описує дані, які використовуються в додатку [6]. Моделі можуть містити логіку валідації даних, проте не повинні визначати логіку відображення цих даних або взаємодії з елементами графічного інтерфейсу.

2. View – представлення графічного інтерфейсу, з яким взаємодіє користувач. Представлення визначає перелік та розміщення графічних елементів керування та логіку взаємодії користувача з ними за допомогою команд.

3. ViewModel – модель представлення – своєрідний посередник між моделлю та її представленням. Модель представлення призначена для переправлення даних, введених через графічний інтерфейс до моделі, та навпаки – при зміні даних в моделі – відобразити ці зміни у графічному інтерфейсі. Також

модель представлення реалізує команди взаємодії з графічним інтерфейсом, наприклад, команду зміни представлення (навігацію).

MVVM призначений для проектування додатків на платформах Windows Presentation Foundation та Microsoft Silverlight.

Windows Presentation Foundation – система для створення графічних програмний додатків, розроблена компанією Microsoft, яка використовує мову XAML для опису графічної складової (компонента View шаблону MVVM) [7].

На відміну від свого попередника – Windows Forms – WPF використовує апаратне прискорення відображення графіки засобами DirectX [8], що дозволяє програмним додаткам, створеним на WPF запускатися та змінювати графічний інтерфейс у відповідності зі змінами значно швидше.

Також платформа Windows Presentation Foundation дозволяє використовувати в додатку відображення 3D-графіки, зв'язування графічних елементів зі стилями та надає можливість автоматичної адаптації застосунку до різних параметрів роздільної здатності екрану пристрою [9-10].

Більшість сучасних додатків тим чи іншим чином пов'язані з веденням даних про певні об'єкти з подальшим збереженням цих даних у базу даних. Цей процес у класичному програмному додатку із графічним інтерфейсом включає в себе такі основні дії, як:

– копіювання даних з керованих об'єктів у елементи керування, де дані можна відображати та редагувати;

– забезпечення копіювання змін, внесених до даних за допомогою елементів керування, назад до керованих об'єктів.

За допомогою механізму прив'язок даних, схема роботи якого наведена на рис. 2.2, WPF значно полегшує процес отримання та надсилання даних між моделлю та представленням, і, таким чином, полегшує роботу з базою даних.

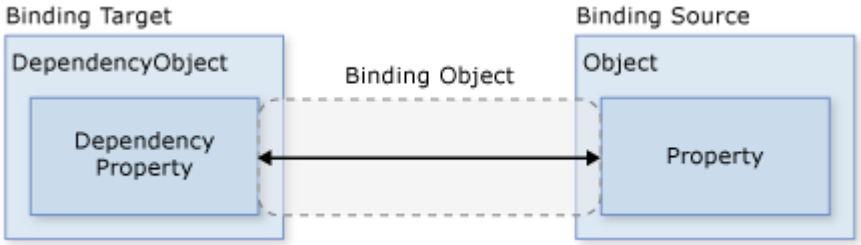


Рис. 2.2. Модель роботи механізму прив’язки даних

Використовуючи клас Binding у файлі розмітки представлення програмного додатку можна виконати зв’язку графічного елемента, наприклад, пункту меню з командою, яка буде виконана при його активації користувачем. Приклад такої операції наведено на рис. 2.3.

```
<Menu Grid.Row="0" Height="25">
    <MenuItem Header="Навігація">
        <MenuItem Command="{Binding GoBackCommand}" Header="На попередню" />
        <MenuItem Command="{Binding GoToMainPageCommand}" Header="На головну" />
    </MenuItem>
</Menu>
```

Рис. 2.3 Прив’язка команди до графічного елемента керування у XAML

2.4. Опис використаних технологій та мов програмування

Програмний додаток було реалізовано мовою програмування C# з використанням технологій WPF.

C# – об’єктно-орієнтована мова програмування високого рівня, створена у 2001 році групою розробників під керуванням Microsoft Research. C# належить до родини C подібних мов програмування та спирається на досвід використання своїх попередників, що дозволяє виключити певні особливості, які прийнято вважати проблемними при розробці програмних додатків, наприклад, множинне успадкування, механізм вказівників тощо.

C# реалізує методологію об’єктно-орієнтованого програмування з її складовими частинами:

– інкапсуляція – приховання від «зовнішнього світу» внутрішньої реалізації компоненту та його даних із наданням доступу до програмних методів керування ним [11];

– поліморфізм – можливість перевизначити методи батьківського класу для отримання результату певної дії похідного класу при різних його реалізаціях;

– абстракція – можливість розробника концентруватись на таких властивостях розроблюваного об’єкта, які необхідні для вирішення поточної задачі;

– успадкування – можливість створення загального класу, який буде базовим для інших та визначатиме спільні для них поля та методи.

Оскільки C# є об’єктно-орієнтована мова програмування – усі функції створюваного додатку повинні втілювати об’єктний підхід. Прикладом такої реалізації може слугувати виконання найпростішої програми – виведення рядку на екран, наведене на рис. 2.4.

```
using System;

class Hello
{
    static void Main()
    {
        Console.WriteLine("Hello, World");
    }
}
```

Рис. 2.4. Приклад програми мовою C#

Усі типи даних у C#, включаючи визначені користувачем типи даних, успадковуються від класу object, та мають спільний набір операцій. Існують два різновиди типів типи значень та посилкові типи. Змінні, які оголошені через тип значень зберігають в собі безпосередньо дані, в той час, як змінні посилальних типів зберігають та оперують посиланнями на необхідні дані – об’єкти [12].

Мова програмування C# надає чимало корисних функцій для полегшення процесу розробки програмного забезпечення:

– функціонал зі збирання сміття автоматично, без участі розробника, вивільняє пам’ять, яка була виділена об’єктами, що більше не використовуються або недоступні;

– обробка виключень дозволяє структурувати виявлення помилок та відновлення після їхнього виправлення;

– підтримка механізму використання лямбда-виразів;

– підтримка асинхронних операцій тощо.

Завдяки особливостям архітектури платформи .NET – цільової платформи мови C#, програми написані нею виконуються у віртуальній системі виконання. Програмний код C# переводиться у проміжну мову, яка відповідає специфікації CLI [13]. Коли програма C# виконується, її збірка розміщується в середовищу CLR – реалізації загальномовної інфраструктури мови, яка є міжнародним стандартом від компанії Microsoft [14]. Середовище CLR, в свою чергу виконує JIT-компіляцію коду мовою IL в команди машинної мови. Окрім того, CLR також керує збором сміття, обробку виключень, керування ресурсами та інше.

Програмний додаток було створено засобами середовища розробки Microsoft Visual Studio Community 2022. Перевагами даної версії середовища розробки є:

– безкоштовність;

– підтримка високорівневих мов програмування, зокрема C#;

– підтримка технології розробки WPF та .NET;

– підтримка технології IntelliSense для автодоповнення змісту програмного рядку;

– можливість перебування внутрішньої структури програмного коду, яка не зачіпає зовнішню поведінку програми [15];

– графічний конструктор для проектів, створених за технологією WPF;

– наявність дизайнеру класів та дизайнеру схеми бази даних;

– можливість розширення функціоналу за рахунок встановлення сторонніх додатків.

Для поєднання програмного додатку із базою даних було використано технологію Entity Framework Core [16] – сучасний модуль зв’язки «об’єкт – база даних», призначений для .NET, та працюючий із СУБД SQLite, MySQL, PostgreSQL та іншими.

Для відображення вмісту бази даних у програмному додатку з використанням EF необхідно у проекті створити клас-модель, який буде відображати вміст таблиці бази даних. Приклад класу моделі наведено на рис. 2.5.

```
public class ApplicationContext : DbContext
{

    public class User
    {

        public int id { get; set; }
        — ссылки
        public string UserName { get; set; }
        — ссылки
        public string Password { get; set; }
        — ссылки
        public int Age { get; set; }
        — ссылки
        public double Height { get; set; }
        — ссылки
        public double Weight { get; set; }
        — ссылки
        public string Gender { get; set; }
        — ссылки
        public bool IsSuperUser { get; set; }


        public User(string userName, string password, int age, double height, double weight, string gender, bool isSuperUser)
        {
            UserName = userName;
            Password = password;
            Age = age;
            Height = height;
            Weight = weight;
            Gender = gender;
            IsSuperUser = isSuperUser;
        }



        public ObservableCollection<UserGoodList> CustomGoods { get; set; } = new ObservableCollection<UserGoodList>();

        public ObservableCollection<UserRestrictionList> CustomRestrictions { get; set; } = new ObservableCollection<UserRestrictionList>();

        public ObservableCollection<GoodBasket> GoodBaskets { get; set; } = new ObservableCollection<GoodBasket>();
    }

    public DbSet<User> Users { get; set; }
```

Рис. 2.5. Приклад створення моделі Entity Framework на основі класу User

Використовуючи технологію Entity Framework звертатися до вмісту бази даних можна безпосередньо в програмному коді дотатку, використовуючи синтаксис Language Integrated Query (LINQ) [17] приклад якого наведено на рис. 2.6.

```
using (ApplicationContext db = new ApplicationContext())
{
    var result = from userGood in db.UserGoodList
                 join goodInShop in db.GoodInShop on userGood.GoodInShopID equals goodInShop.id
                 join good in db.Goods on goodInShop.GoodId equals good.id
                 join goodShopState in db.GoodShopState on goodInShop.id equals goodShopState.GoodInShopID
                 join goodState in db.GoodState on goodShopState.GoodStateID equals goodState.id
                 where userGood.UserID == user.id
                 select new
                 {
                     GoodInShopID = goodInShop.id,
                     GoodName = good.GoodName,
                     GoodPrice = goodInShop.GoodPrice,
                     GoodAmount = goodInShop.GoodAmount,
                     GoodUnits = goodInShop.GoodUnits,
                     GoodCarbohydrates = goodState.Carbohydrates,
                     GoodID = goodInShop.GoodId
                 };
}
```

Рис. 2.6. Приклад запиту до бази даних за допомогою LINQ

У якості системи управління базами даних було обрано PostgreSQL – об’єктно-реляційну СУБД, розроблену у 1996 році Майклом Стоунбрейкером в Каліфорнійському університеті [18].

PostgreSQL було обрано у якості системи управління базами даних через ряд переваг:

– підтримка стандартів мови SLQ;
– відсутність обмежень розміру бази даних;
– відсутність обмежень кількості записів в таблиці;
– відсутність обмежень індексів в таблиці;
– надання максимального розміру таблиці рівному 32 ТБ;
– максимальний розмір одного запису 1,6 ТБ;
– максимальний розмір одного поля таблиці – 1 ГБ;
– високий ступінь надійності.

Окрім того, PostgreSQL відповідає стандарту ANSI-SQL:2008 [19] та задовольняє вимоги ACID (атомарність, узгодженість, ізольованість та надійність) [20]. Система має різноманітний функціонал (перевірочні обмеження, каскадні зовнішні ключі, обмеження NOT NULL тощо) із забезпечення збереження тільки коректних даних [21].

2.5. Опис структури програми та алгоритмів її функціонування

Структура розробленого проекту наведена на рис. 2.7.

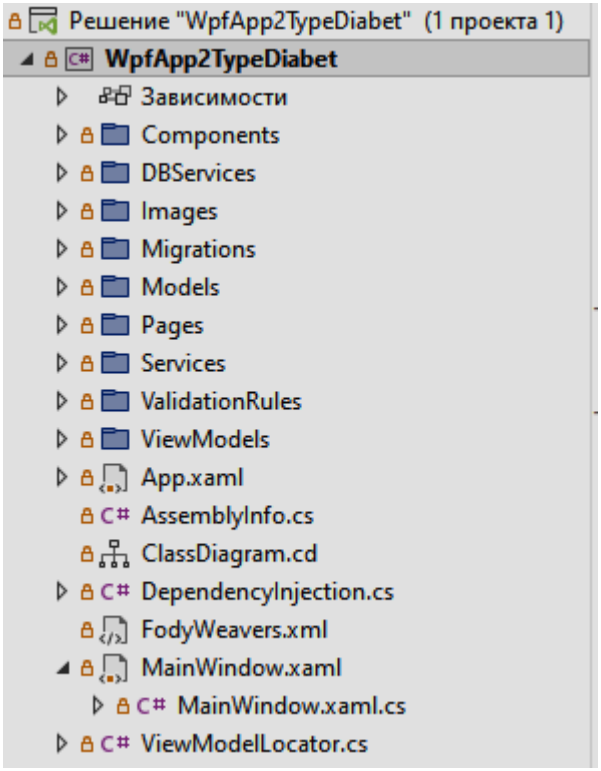


Рис. 2.7. Структура проекту

Тека Components призначена для зберігання створеного користувальницького компоненту BindablePasswordBox – графічного контролеру для введення паролю із можливістю отримання його значення в моделі представлення.

Тека DBServices призначена для збереження класу організації підключення до бази даних.

Папка Images зберігає іконки для програмного продукту.

Папка Migrations зберігає міграційні структурні зміни бази даних – створення, редагування, видалення.

Тека Models містить створені моделі програмної системи, які, в свою чергу містять в собі дані про об’єкти – користувачів, продукти, продуктові кошики тощо.

Папка Pages містить в собі усі представлення програмного додатку, за якими здійснюється навігація – головна сторінка, сторінка авторизації, реєстрації, додавання нового продукту тощо.

У теці Services містяться допоміжні класи-сервіси для керування даними.

Правила перевірки даних на коректність знаходяться в теці ValidationRules та використовуються на багатьох сторінках програмного додатку (сторінка авторизації, реєстрації, додавання обмеження щодо споживання продукту тощо).

Тека ViewModels містить моделі представлень програмного додатку та керує передачею даних між сторінками, навігацією, перевірками даних на коректність тощо.

База даних складається з одинадцяти пов’язаних між собою таблиць:

- goods — «Продукти (товари)»;
- categories — «Категорія продукту (товару)»;
- goodInShop — «Продукт (товар) в магазині»;
- goodState — «Стан продукту (товару)»;
- restriction — «Обмеження щодо споживання продукту (товару)»;
- goodShopState — «Стан продукту (товару) в магазині»;
- goodBasket — «Продуктовий кошик»;
- users — «Користувачі»;
- userGoodList — «Список продуктів (товарів) користувача»;
- userRestrictionList — «Список обмежень користувача»;
- goodInBasket — «Продукт (товар) у продуктовому кошику».

Схема створеної бази даних, яка складається з 11 таблиць наведена на рис. 2.8.

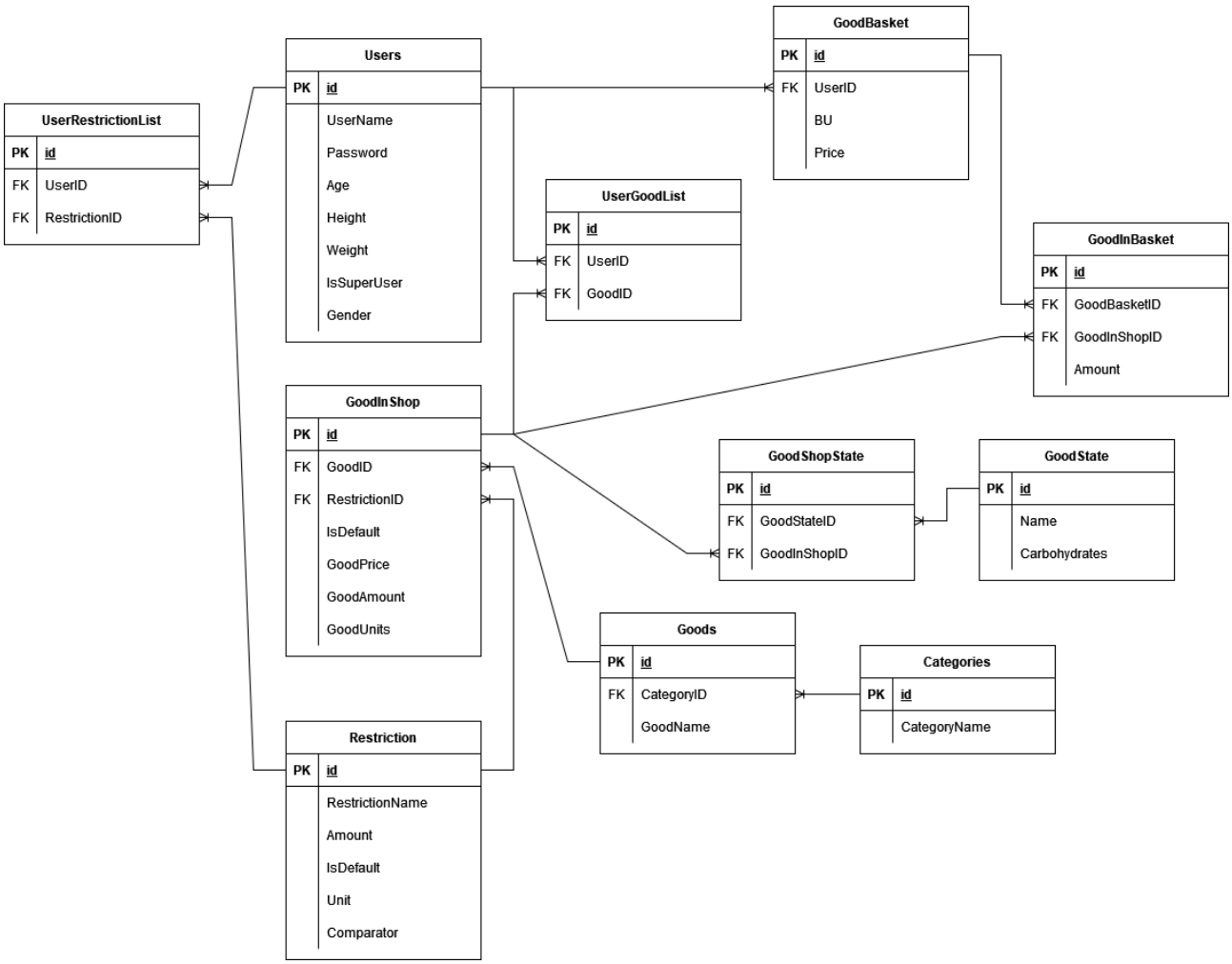


Рис. 2.8. ER діаграма бази даних

Структура таблиць бази даних описана в таблицях 2.1 – 2.11.

Таблиця 2.1

Користувачі

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
UserName	Логін користувача	text	

Закін. Табл. 2.1

Назва поля	Призначення поля	Тип даних	Ключ
Password	Пароль користувача	text	
Age	Вік	int	
Height	Зріст	double	
Weight	Вага	double	
Gender	Стать	text	
IsSuperUser	Показник, чи є користувач адміністратором	boolean	

Таблиця 2.2

Категорія продукту (товару)

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
CategoryName	Назва категорії товару	text	

Таблиця 2.3

Продукти (товари)

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
GoodName	Назва товару	text	
CategoryID	Ідентифікатор категорії товару	int	Foreign key

Таблиця 2.4

Продукт (товар) в магазині

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
GoodId	Ідентифікатор продукту (товару)	int	Foreign key
IsDefault	Показник, чи є продукт (товар) стандартним	boolean	
GoodPrice	Ціна продукту (товару)	double	

Закін. табл. 2.4

Назва поля	Призначення поля	Тип даних	Ключ
GoodAmount	Кількість продукту (товару)	double	
GoodUnits	Одиниці виміру	text	
RestrictionID	Ідентифікатор обмеження щодо споживання	int	Foreign key

Таблиця 2.5

Стан продукту (товару) в магазині

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
GoodStateID	Ідентифікатор стану продукту (товару)	int	Foreign key
GoodInShopID	Ідентифікатор продукту (товару) в магазині	int	Foreign key

Таблиця 2.6

Стан продукту (товару)

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
Name	Назва стану продукту (товару)	text	
Carbohydrates	Кількість хлібних одиниць в продукті (товарі)	double	

Таблиця 2.7

Список продуктів (товарів) користувача

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
GoodInShopID	Ідентифікатор продукту (товару) в магазині	int	Foreign key
UserID	Ідентифікатор користувача	int	Foreign key

Таблиця 2.8

Список обмежень користувача

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
RestrictionID	Ідентифікатор обмеження щодо споживання продукту (товару)	int	Foreign key
UserID	Ідентифікатор користувача	int	Foreign key

Таблиця 2.9

Продукт (товар) у продуктовому кошику

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
GoodBasketID	Ідентифікатор продуктового кошику	int	Foreign key
Amount	Кількість товару (товару)	double	

Закін. табл. 2.9

Назва поля	Призначення поля	Тип даних	Ключ
GoodInShopID	Ідентифікатор продукту (товару) в магазині	int	Foreign key

Таблиця 2.10

Продуктовий кошик

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
UserID	Ідентифікатор користувача	int	Foreign key
BU	Загальна кількість хлібних одиниць	double	
Price	Загальна ціна кошику	double	

Таблиця 2.11

Обмеження щодо споживання продукту (товару)

Назва поля	Призначення поля	Тип даних	Ключ
id	Унікальний ідентифікатор запису в базі даних	int	Primary key
RestrictionName	Назва обмеження	text	
Amount	Кількість продукту	double	
IsDefault	Показник, чи є обмеження стандартним	boolean	
Comparator	Тип обмеження	text	
Unit	Одиниці виміру	text	

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Створений програмний додаток отримує на вхід та зберігає в базі даних інформацію про такі об’єкти:

– користувач (вік, стать, вага, зріст, рівень фізичного навантаження);

– продукт (назва, ціна, кількість, одиниці виміру, кількість вуглеводів у 100 грамах продукту, категорія);

– обмеження щодо споживання продукту (назва, кількість продукту, одиниці виміру, тип обмеження);

– дані для оптимізації раціону харчування (максимальна вартість продуктового кошику, часовий проміжок).

У якості вихідних даних виступає створюваний вміст продуктового кошику згідно із наявними вимогами та обмеженнями:

- оптимізований список продуктів харчування;
- загальна вартість продуктового кошику;
- загальна кількість хлібних одиниць.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Розроблюване програмне забезпечення створювалося, та призначене для ЕОМ за наступними технічними характеристиками:

- оперативний запам’ятовувальний пристрій об’ємом 2 ГБ або більше;
- процесор з тактовою частотою 1 ГГц або вище;
- жорсткий диск об’ємом не менше 20 ГБ;
- наявність маніпулятора-миші та клавіатури;
- монітор SVGA.

2.7.2. Використані програмні засоби

Під час розробки програмного забезпечення були використані наступні програмні засоби:

- Microsoft Visual Studio;
- pgAdmin 4;
- PostrgeSQL;
- GitHub.

Використано програмні можливості, надані мовою програмування C# та платформою .Net Core. Візуальна частина програмного додатку виконана за допомогою можливостей технології WPF.

2.7.3. Виклик та завантаження програми

Оскільки робота з програмним додатком передбачає використання бази даних PostgreSQL – програмний додаток необхідно інсталювати. Для цього необхідно запустити файл «InstallWpfApp2TypeDiabet», та слідувати інструкціям інсталятора.

Після встановлення для запуску програмного додатку необхідно запустити на виконання файл «WpfApp2TypeDiabet.exe». Програмний додаток не потребує підключення до мережі для коректної роботи.

2.7.4. Опис інтерфейсу користувача

Реєстрація та авторизація в додатку

Під час запуску програмного додатку користувачеві відображається вікно, зображене на рис. 2.9.

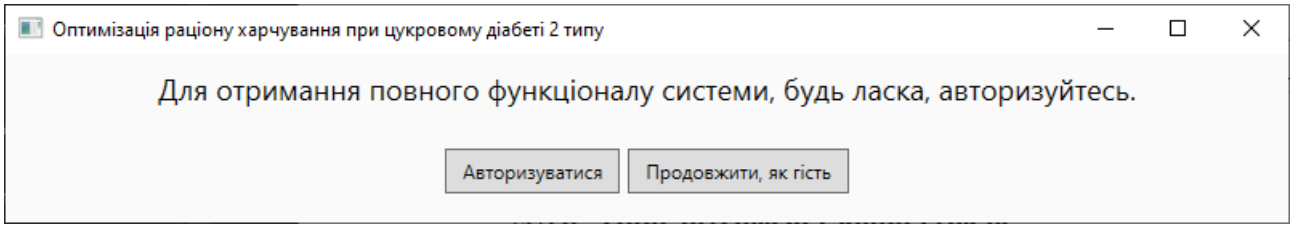


Рис. 2.9. Вікно вибору функціоналу системи

Якщо користувач обере пункт «Продовжити, як гість» - йому буде наданий обмежений функціонал системи, який включає перегляд стандартних продуктів харчування, обмежень щодо їхнього споживання та проведення обчислень з оптимізації раціону харчування.

Якщо користувач обере пункт «Авторизуватися» – він перейде на сторінку, зображену на рис. 2.10.

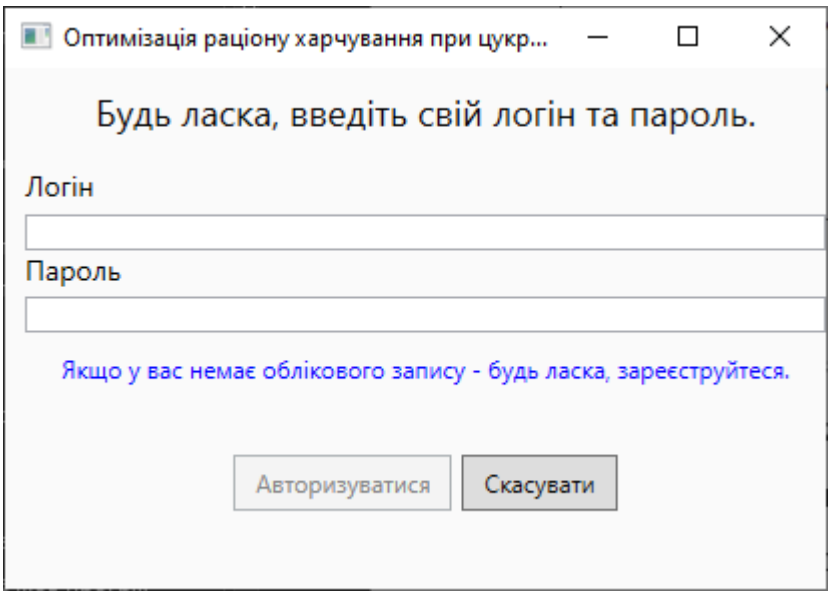


Рис. 2.10. Зовнішній вигляд форми авторизації

Користувачеві пропонується авторизуватися у системі шляхом введення свого логіну та паролю. Якщо додаток запускається вперше, і у користувача немає облікового запису, проте він бажає отримати повний функціонал програмного додатку – користувач має можливість пройти процес реєстрації облікового запису, натиснувши на відповідне посилання. В цьому випадку відкриється сторінка, зображена на рис. 2.11.

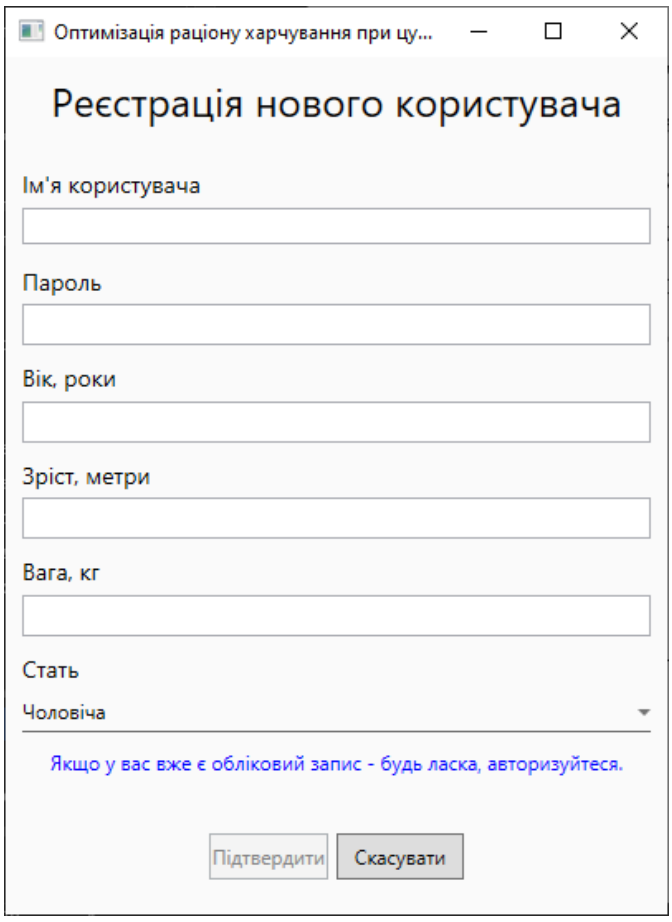


Рис. 2.11. Зовнішній вигляд сторінки реєстрації облікового запису

На формі реєстрації користувачеві необхідно ввести усі відповідні дані. Окрім того, ці дані повинні відповідати певному формату:

– ім’я користувача повинно буди написано латинськими літерами та мати хоча б одну велику літеру;

– пароль повинен містити щонайменше 8 символів, серед яких повинно бути не менше однієї цифри, спеціального символу та великої літери;

– вік користувача повинен бути числовим значенням, не перевищувати значення в 200 років та не дорівнювати нулю;

– зріст користувача може бути цілочисельним або дійсним значенням, не перевищуючим 3 метри;

– вага користувача повинна бути цілочисельним або дійсним значенням, не перевищуючим 1000 кілограм.

У випадку, коли користувач не додержується вимог до вхідних даних – йому буде видано відповідне повідомлення, приклад якого наведено на рис. 2.12.

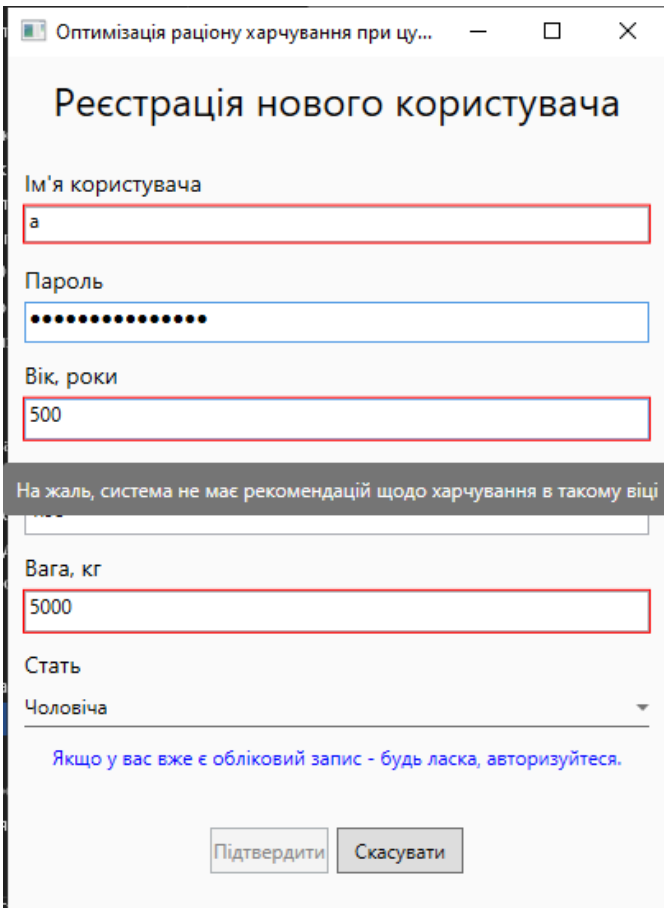


Рис. 2.12. Приклад зовнішнього вигляду повідомлення про помилку при введенні даних

Якщо користувач ввів коректні дані для реєстрації, проте введене ім’я користувача вже зареєстровано в системі – користувачеві буде видано відповідне повідомлення, зображене на рис. 2.13.

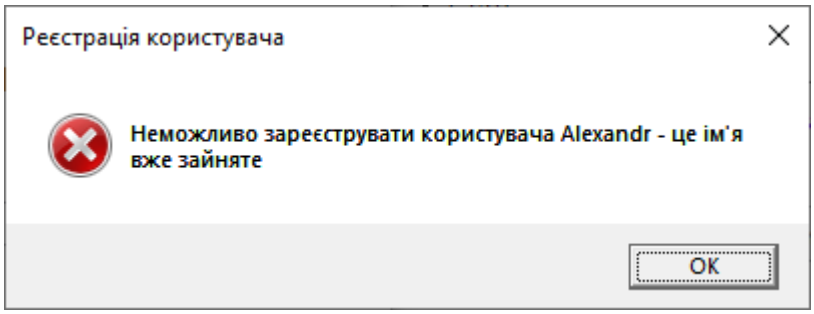


Рис. 2.13. Приклад повідомлення про неможливість авторизації

Якщо користувач натисне на кнопку «Скасувати» в процесі реєстрації – йому буде видано вікно-підтвердження, зображене на рис. 2.14.

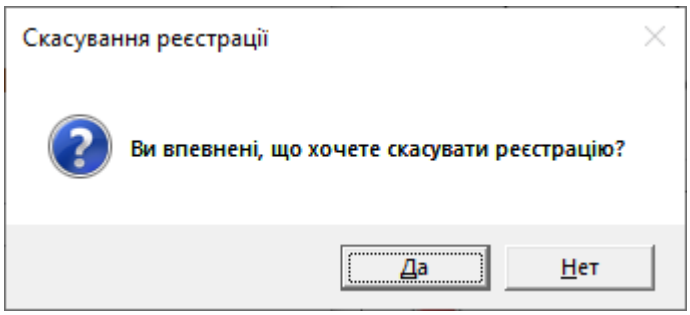


Рис. 2.14. Зовнішній вигляд запиту на підтвердження скасування реєстрації

При підтвердженні наміру користувач повернеться до сторінки авторизації.

Коли користувач успішно зареєстрував свій обліковий запис – йому буде видано відповідне повідомлення, зображене на рис. 2.15 та направлено на сторінку авторизації, де необхідно буде ввести ім’я свого облікового запису та пароль.

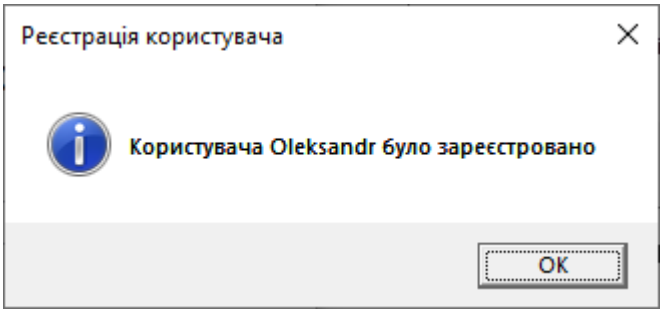


Рис. 2.15. Повідомлення про успішну реєстрацію в додатку

Якщо в процесі авторизації користувач введе хибні дані – йому буде видано відповідне повідомлення, зображене на рис. 2.16, а процес авторизації буде зупинений.

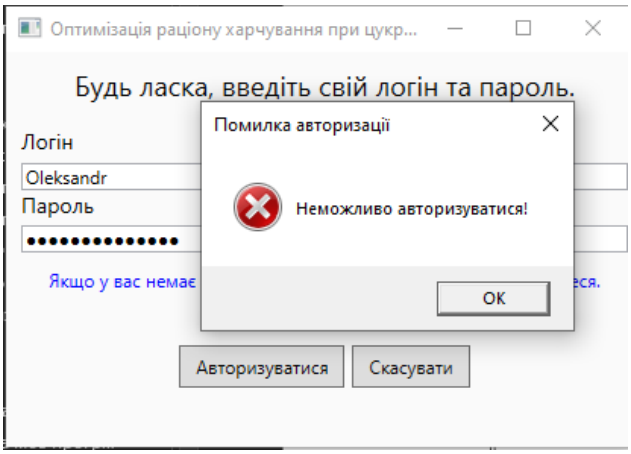


Рис. 2.16. Результат помилкової спроби авторизації

При успішній авторизації користувач потрапить на головну сторінку додатку, зображену на рис. 2.17, звідки зможе перейти до взаємодії з функціоналом додатку з перегляду та створення продуктів харчування, обмежень, продуктових кошиків, перегляду та зміни особистих даних, вийти з облікового запису або завершити роботу додатку.

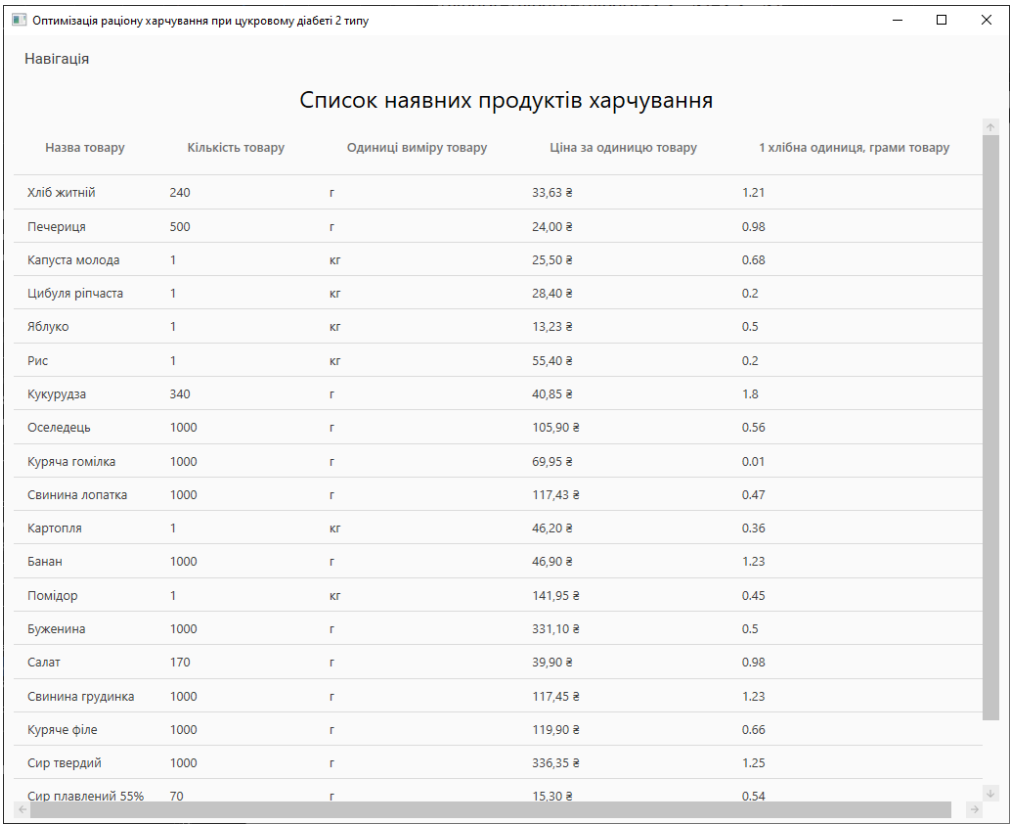
Рис. 2.17. Головна сторінка додатку

Перегляд списку продуктів харчування та їх додавання і видалення

Для того, щоб переглянути список продуктів харчування користувачеві необхідно на головній формі обрати пункт меню «Продукти» (рис. 2.18), та обрати тип продуктів харчування, який він хоче переглянути – «Стандартні» чи «Користувальницькі».

Рис. 2.18. Зміст пункту меню «Продукти»

Якщо користувач обере для перегляду стандартний тип продуктів харчування – він потрапить на сторінку, зображену на рис. 2.19, на якій міститься список усіх наявних стандартних продуктів харчування. Змінювати цей список бідь-яким чином користувачеві не дозволено.



Назва товару	Кількість товару	Одиниці виміру товару	Ціна за одиницю товару	1 хлібна одиниця, грами товару
Хліб житній	240	г	33,63 ₴	1.21
Печериця	500	г	24,00 ₴	0.98
Капуста молода	1	кг	25,50 ₴	0.68
Цибуля ріпчаста	1	кг	28,40 ₴	0.2
Яблуко	1	кг	13,23 ₴	0.5
Рис	1	кг	55,40 ₴	0.2
Кукурудза	340	г	40,85 ₴	1.8
Оселедець	1000	г	105,90 ₴	0.56
Куряча гомілка	1000	г	69,95 ₴	0.01
Свинина лопатка	1000	г	117,43 ₴	0.47
Картопля	1	кг	46,20 ₴	0.36
Банан	1000	г	46,90 ₴	1.23
Помідор	1	кг	141,95 ₴	0.45
Буженина	1000	г	331,10 ₴	0.5
Салат	170	г	39,90 ₴	0.98
Свинина грудинка	1000	г	117,45 ₴	1.23
Куряче філе	1000	г	119,90 ₴	0.66
Сир твердий	1000	г	336,35 ₴	1.25
Сир плавлений 55%	70	г	15,30 ₴	0.54

Рис. 2.19. Сторінка перегляду стандартного списку продуктів харчування

За бажанням, користувач може повернутися до головної сторінки, скориставшись пунктом меню «Навігація».

Якщо користувач обере власний тип продуктів для перегляду – він потрапить на сторінку, зображену на рис. 2.20.

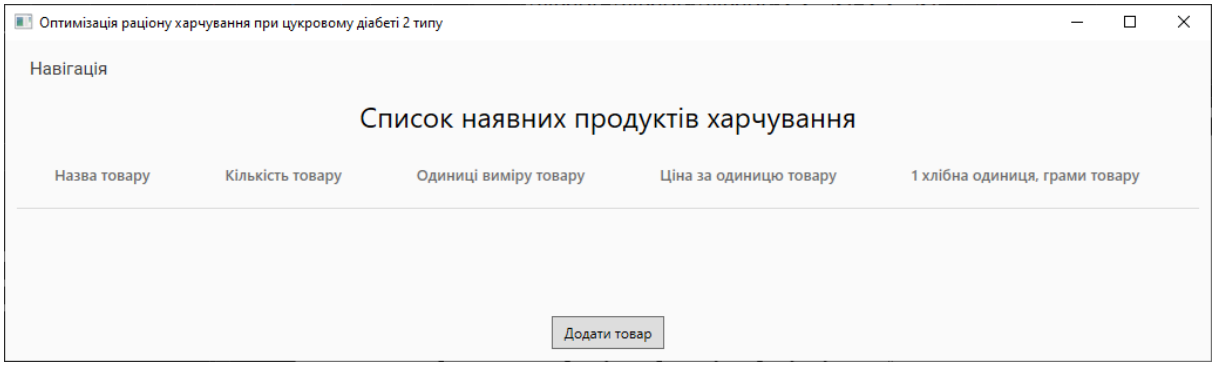


Рис. 2.20. Сторінка перегляд користувальницького списку продуктів харчування

З цієї сторінки користувач може повернутися на головну сторінку додатку, або перейти до сторінки додавання даних про новий продукт (рис. 2.21) харчування шляхом натискання на кнопку «Додати товар».

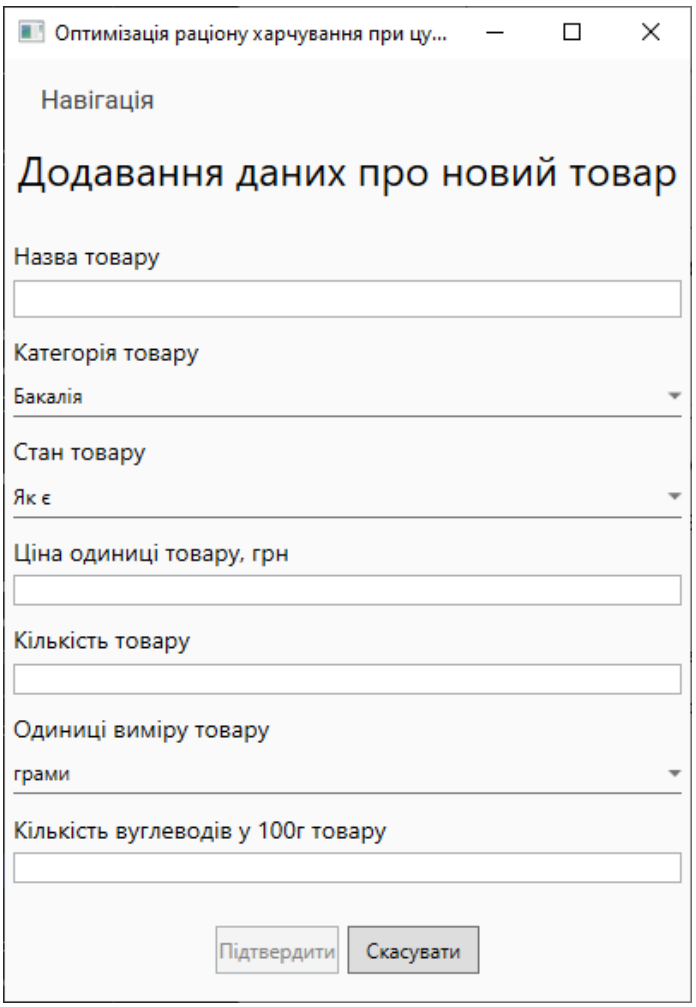


Рис. 2.21. Зовнішній вигляд форми додавання даних про продукт харчування

Дані, що будуть введені користувачем повинні відповідати певним критеріям:

– назва товару повинна бути не коротше двох символів;

– ціна одиниці товару не може дорівнювати нулю та має бути цілочисельним або дійсним значенням;

– кількість товару не може дорівнювати нулю та має бути цілочисельним або дійсним значенням;

– кількість вуглеводів у 100 г товару може бути цілочисельним або дійсним значенням.

У випадку, якщо користувач введе некоректні дані – йому буде виведено відповідні попередження. Після успішного додавання товару буде видано відповідне повідомлення-підтвердження, наведене на рис. 2.22.

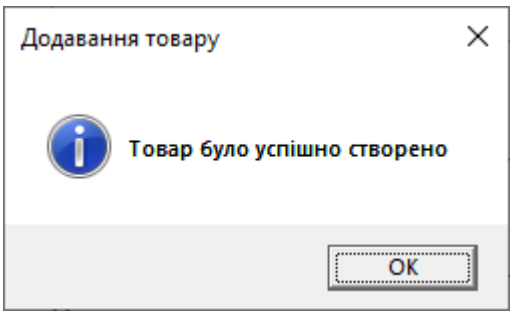


Рис. 2.22. Повідомлення-результат про додавання товару

Після додавання усіх необхідних товарів користувач може повернутися до форми перегляду списку товарів (рис. 2.23).

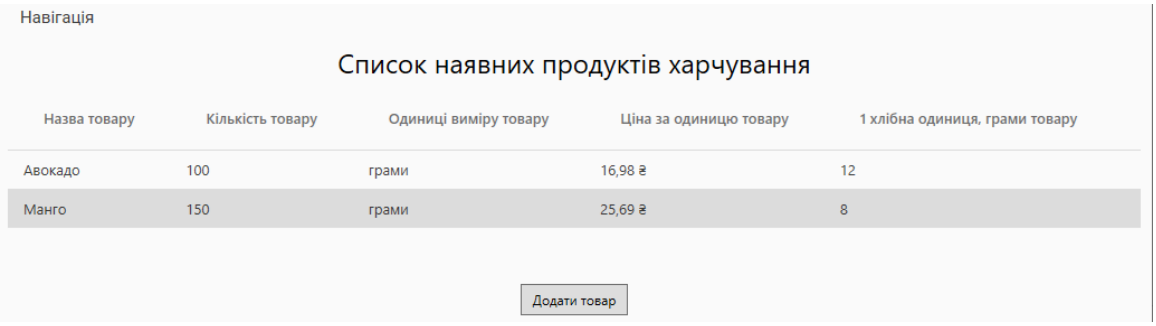


Рис. 2.23. Перегляд користувальницького списку товарів

У списку продуктів користувач може викликати контекстне меню для товару, та видалити його. При цьому буде виведено повідомлення із запитом підтвердження намірів, зображене на рис. 2.24.

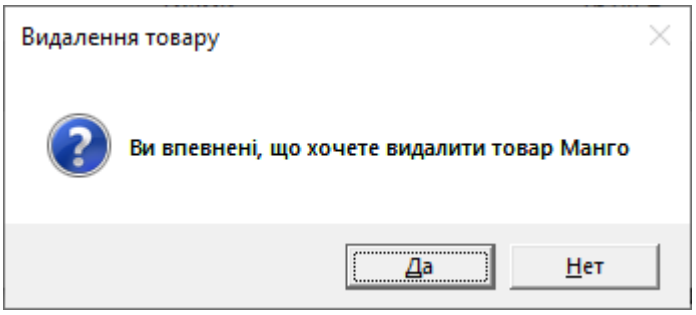


Рис. 2.24. Запит на підтвердження видалення товару

Для перегляду обмежень щодо споживання користувачеві необхідно з головної сторінки додатку перейти за пунктом меню «Обмеження» та обрати підпункт «Стандартні обмеження». Це призведе до відображення сторінки зі стандартними обмеженнями продуктів харчування, зображеної на рис. 2.25.

Навігація

Список наявних обмежень для продуктів харчування

Назва товару	Назва обмеження	Тип обмеження	Кількість	Одиниці виміру
Хліб житній	Стандартне обмеження	Менше або дорівнює	250	г
Печериця	Стандартне обмеження	Менше або дорівнює	250	г
Капуста молода	Стандартне обмеження	Менше або дорівнює	250	г
Цибуля ріпчаста	Стандартне обмеження	Менше або дорівнює	250	г
Яблуко	Стандартне обмеження	Менше або дорівнює	250	г
Рис	Стандартне обмеження	Менше або дорівнює	250	г
Кукурудза	Стандартне обмеження	Менше або дорівнює	250	г
Молоко топлене	Обмеження 1	Менше ніж	1000	мл
Сир плавлений 55%	Стандартне обмеження	Менше або дорівнює	250	г
Сир твердий	Стандартне обмеження	Менше або дорівнює	250	г
Куряче філе	Стандартне обмеження	Менше або дорівнює	250	г
Свинина грудинка	Стандартне обмеження	Менше або дорівнює	250	г
Свинина лопатка	Стандартне обмеження	Менше або дорівнює	250	г

Рис. 2.25. Перегляд стандартних обмежень щодо споживання продуктів харчування

Перегляд та редагування особистих даних користувача

Для перегляду особистих даних користувачеві, на головні сторінці додатку необхідно обрати пункт «Обліковий запис» та перейти за пунктом «Профіль» (рис. 2.26).

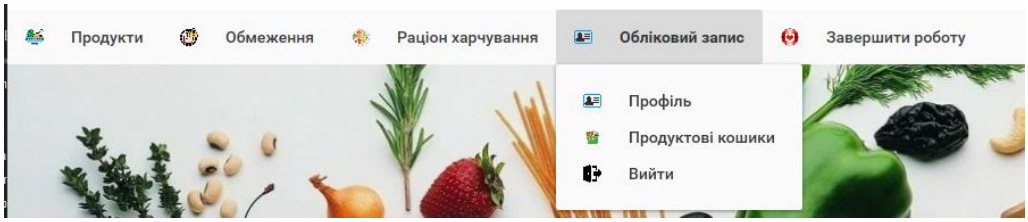


Рис. 2.26. Навігація до облікового запису

Після цього користувачеві відобразиться сторінка з його даними, зображена на рис. 2.27.

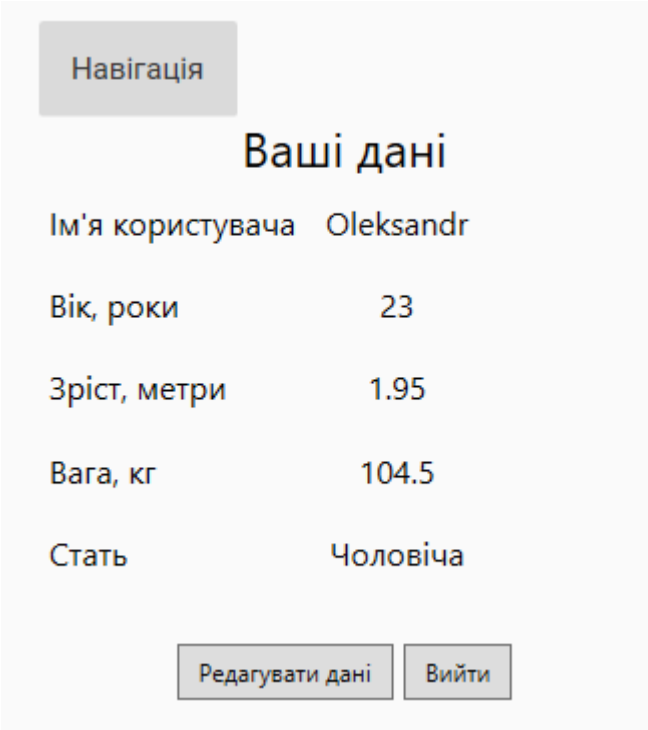


Рис. 2.27. Сторінка перегляду особистих даних користувача

Якщо користувач має намір змінити особисті дані він може перейти на форму редагування (рис. 2.28) шляхом натискання на кнопку «Редагувати дані».

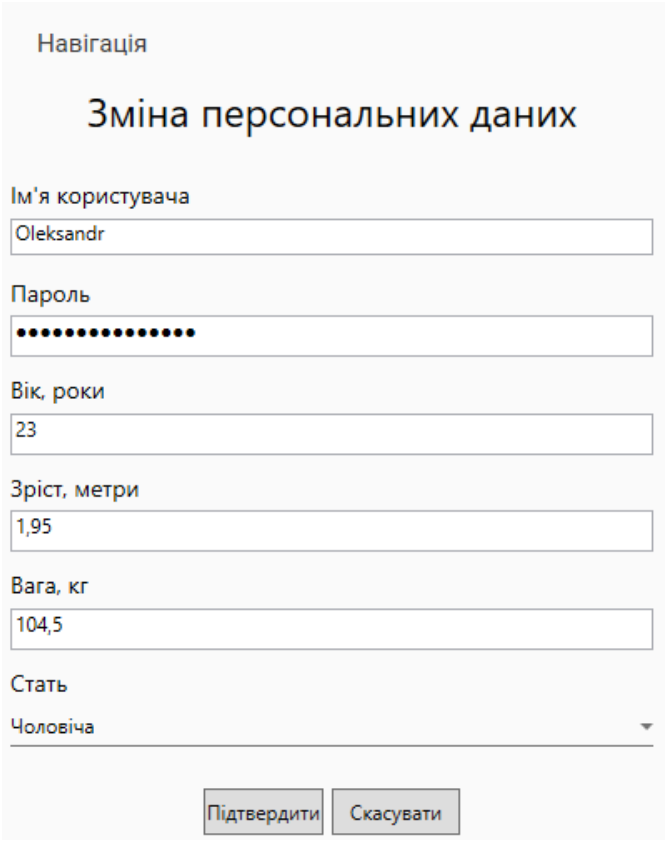


Рис. 2.28. Форма редагування даних користувача

Поля введенні інформації аналогічні полям форми реєстрації облікового запису та мають відповідні вимоги до змісту та формату інформації.

При скасуванні процесу оновлення даних користувачеві буде видано запит на підтвердження дії. При успішному редагуванні – відповідне повідомлення.

Виконання обчислень з оптимізації раціону харчування

Для оптимізації продуктового кошику користувачеві необхідно у головному меню (рис. 2.29) обрати пункт «Раціон харчування» та підпункт «Оптимізувати раціон харчування»

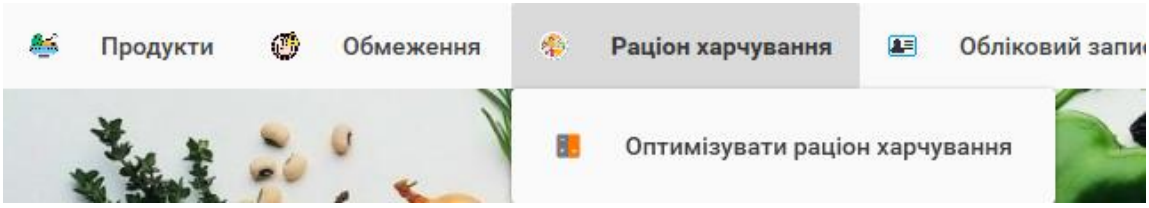


Рис. 2.29. Навігація до функціоналу з оптимізації раціону харчування

На сторінці, що відкрилася (рис. 2.30) користувачеві необхідно підтвердити особисту інформацію та вказати рівень фізичної активності.

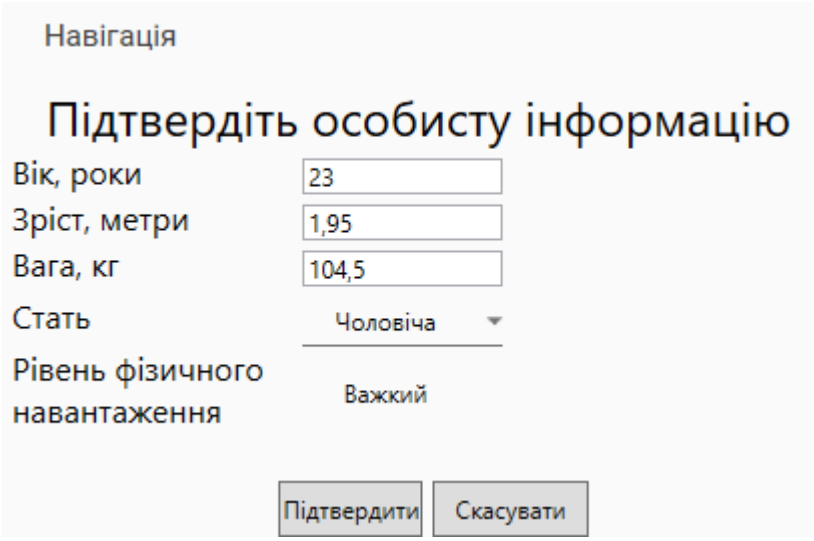


Рис. 2.30. Зовнішній вигляд сторінки підготовки до оптимізації раціону харчування

На наступній відкритій сторінці, зображеній на рис. 2.31, користувачеві необхідно обрати список продуктів, споживання яких він хоче оптимізувати,

вказати максимальну ціну цього продуктового кошику та термін на який будуть виконані обчислення.

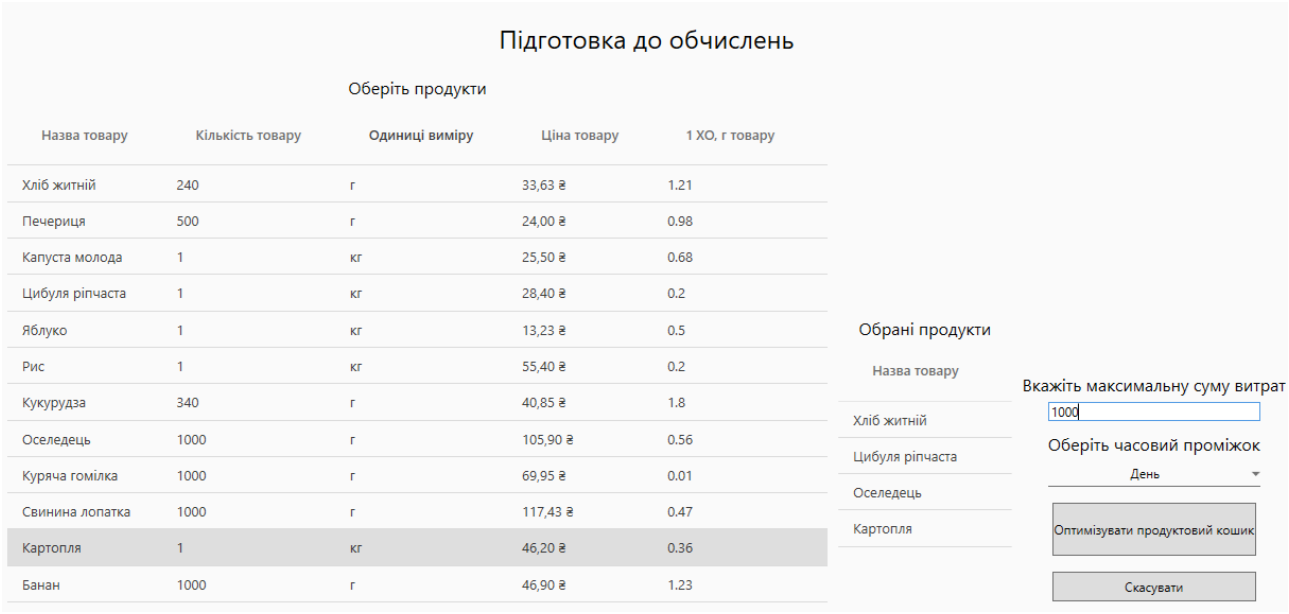


Рис. 2.31. Вікно вибору продуктів та обмежень для проведення оптимізації

Після завершення обчислень користувачеві буде надано список продуктів, який максимально точно відповідає наданим вимогам (рис. 2.32) із кількістю товару, загальною кількістю хлібних одиниць та наявними обмеженнями.

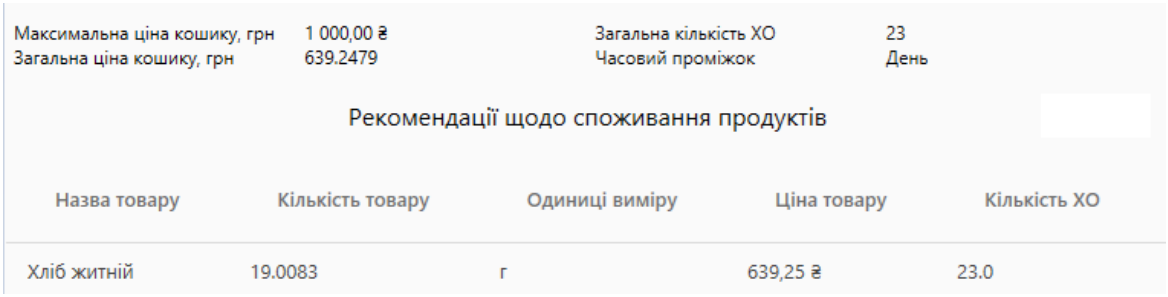


Рис. 2.32. Результати обчислень

За бажанням користувач може зберегти результати обчислень. Для цього необхідно обрати пункт меню «Кошик» на формі перегляду результатів обчислень і підпункт «Зберегти». Після цього відобразиться список наявних результатів оптимізації, наведений на рис. 2.33.

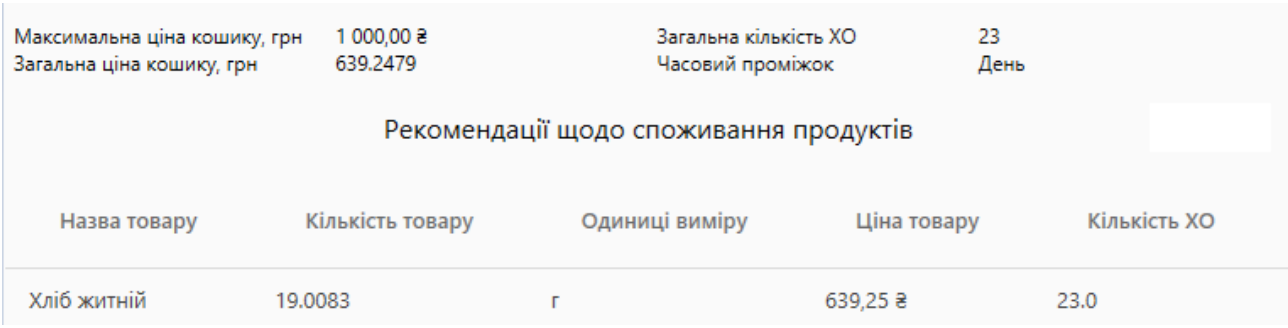


Рис. 2.33. Перегляд нещодавніх обчислень

Додавання стандартних продуктів харчування

Для додавання стандартних продуктів харчування необхідно володіти правами адміністратора додатку.

Після авторизації в ролі адміністратора користувачеві буде доступно головне вікно додатку, наведене на рис. 2.34.

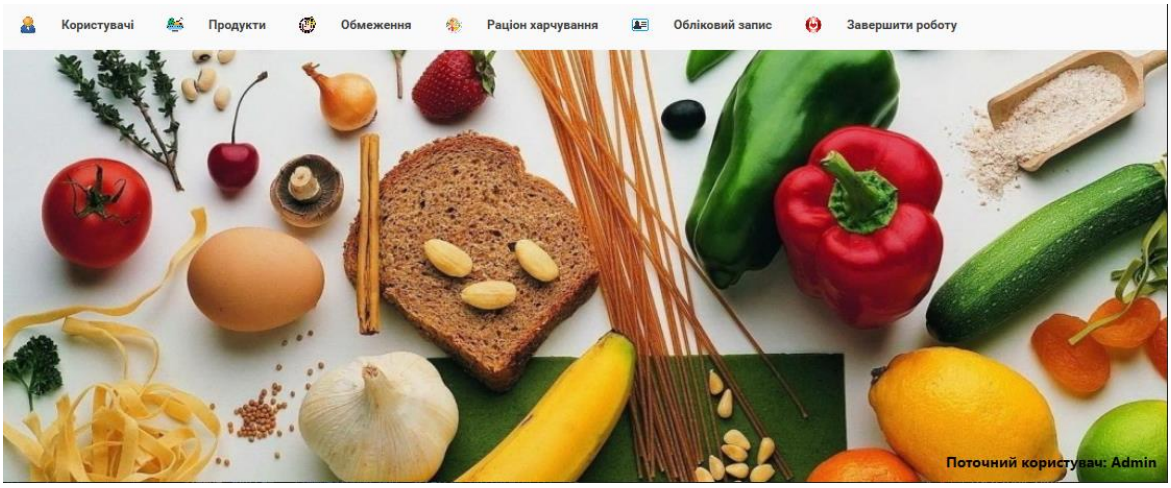


Рис. 2.34. Приклад головного вікна додатку для користувача-адміністратора

Для додавання даних про продукт харчування необхідно перейти за пунктом головного меню «Продукти» та обрати підпункт «Додати продукт». Після цього з’явиться сторінка, зображена на рис. 2.35.

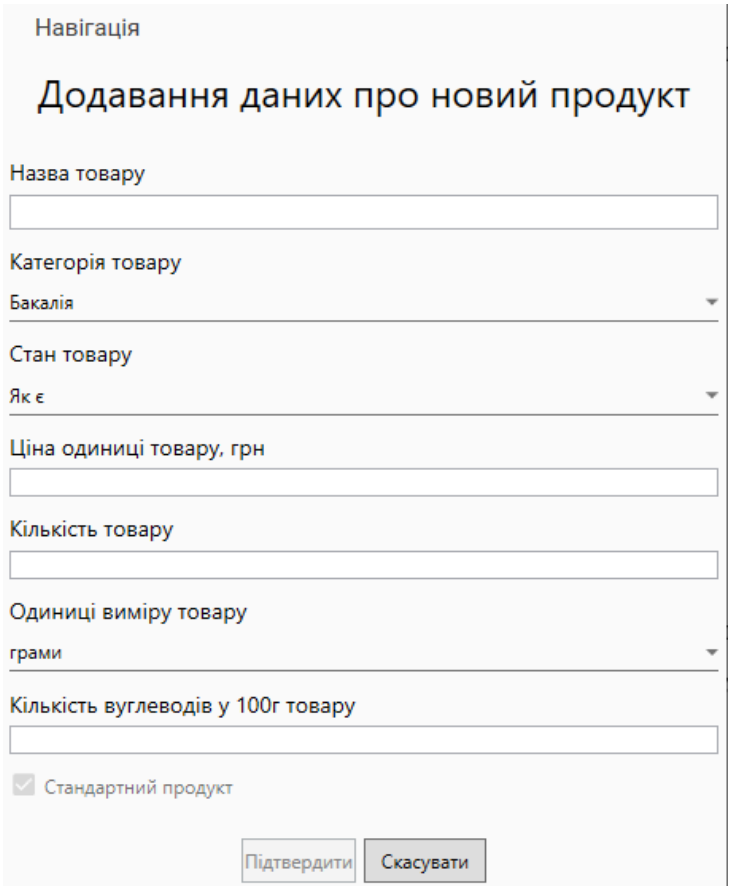


Рис. 2.35. Приклад форми додавання стандартного продукту

Поля введення та формат даних аналогічні відповідним для форми додавання даних про користувальницьких продукт.

Після підтвердження додавання інформації про продукт переглянути її можна у списку стандартних продуктів харчування, зображеному на рис. 2.36.

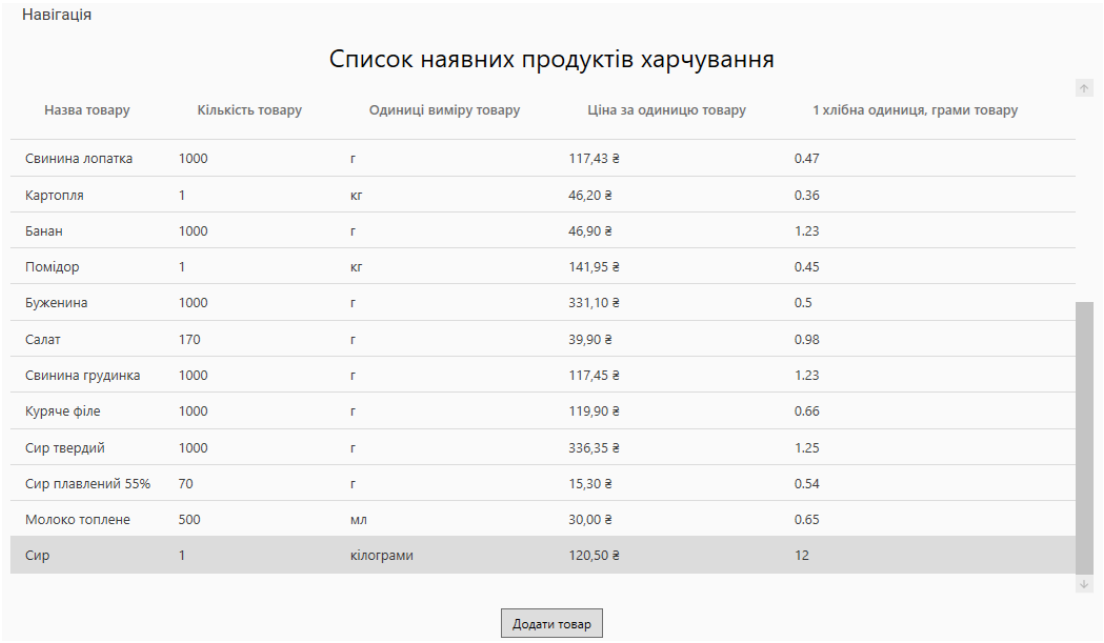


Рис. 2.36. Результат додавання адміністратором стандартного продукту «Сир»

Перегляд та видалення даних про користувачів

Для перегляду списку наявних користувачів системи необхідно із головного вікна за пунктом меню «Користувачі» перейти на сторінку перегляду, зображену на рис. 2.37, на якій буде відображено ім’я користувача, кількість його користувальницьких продуктових товарів, обмежень та продуктових кошиків.

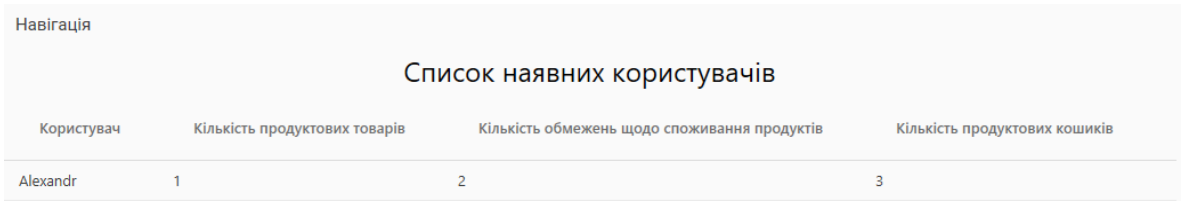


Рис. 2.37. Перегляд списку користувачів програмного додатку

За необхідності, адміністратор може видалити обліковий запис користувача, для чого необхідно викликати контекстне меню для запису в списку та обрати команду «Видалити». У повідомленні – підтвердженні (рис. 2.38) обрати потрібний варіант щодо наміру видалити обліковий запис користувача.

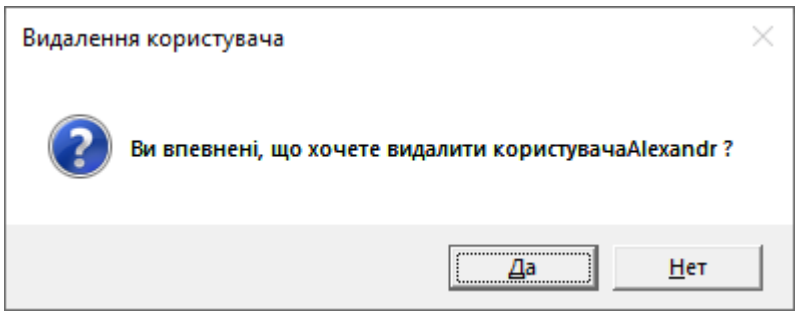


Рис. 2.38. Запит підтвердження видалення користувача

Після успішного видалення облікового запису буде видано відповідне повідомлення, зображене на рис. 2.39.

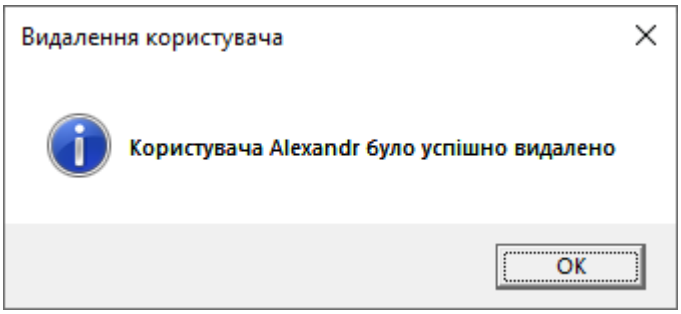


Рис. 2.39. Повідомлення – результат про видалення облікового запису користувача

Вихід з облікового запису

За необхідністю, користувач може вийти зі свого облікового запису за допомогою пункту меню «Обліковий запис» - «Вийти», або «Обліковий запис» - «Профіль» - «Вийти». При спробі вийти з облікового запису користувачеві буде необхідно підтвердити свої наміри у формі, зображеній на рис. 2.40.

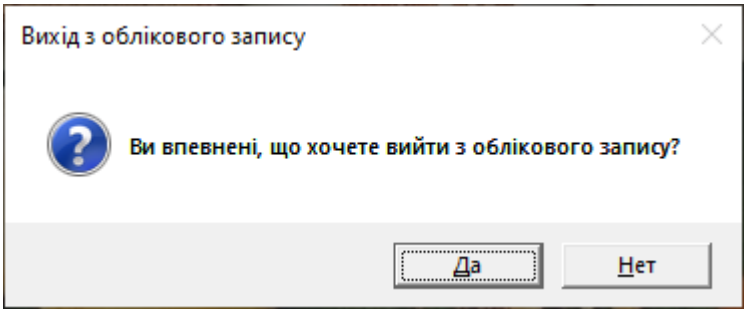


Рис. 2.40. Запит на підтвердження виходу з облікового запису

У разі підтвердження користувачеві відобразиться форма запиту до авторизації.

Завершення роботи програми

Для завершення роботи програми необхідно на головній формі обрати пункт меню «Завершення роботи», після чого ствердно відповісти на підтвердження намірів, зображене на рис. 2.41.

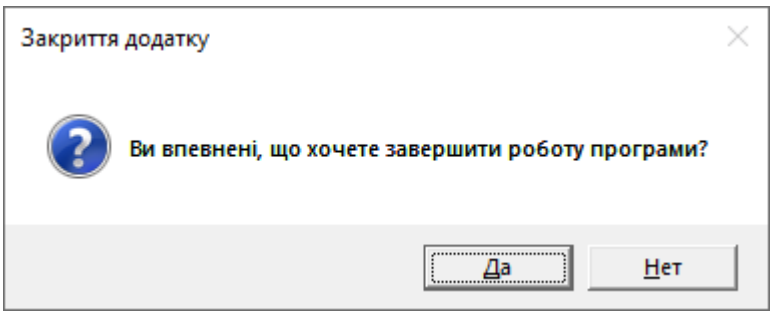


Рис. 2.41. Запит на завершення роботи програми

Приклад результатів обчислень

Результат оптимізації продуктового кошику (рис. 2.42) для користувача з наступними параметрами та вимогами:

– стать – чоловіча;

– рівень фізичного навантаження – низький (кількість ХО визначена в табл. 1.2);

– максимальна вартість продуктового кошику – 200 грн;

– продукти в кошику – хліб, оселедець (рівно 200 грамів), печериці (рівно 300 грамів), цибуля, куряче філе;

– часовий проміжок – один день.

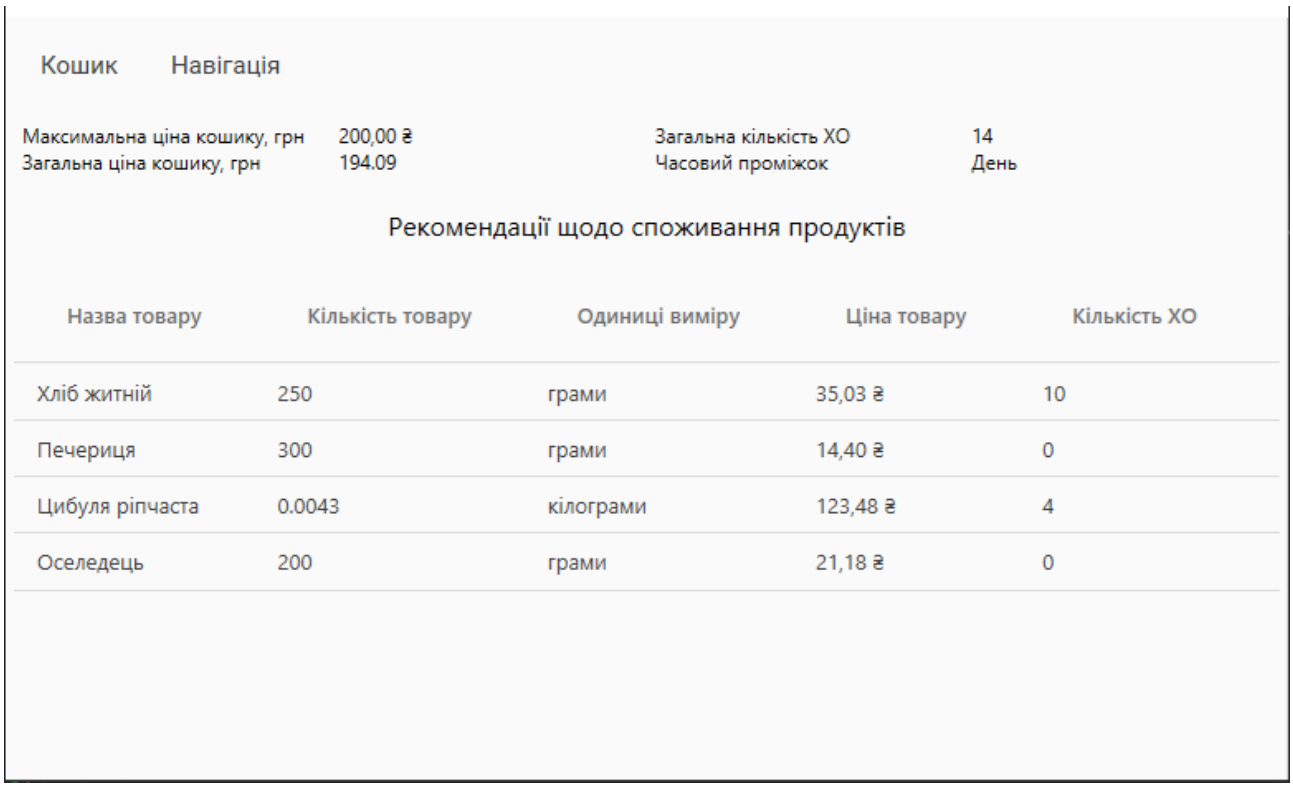


Рис. 2.42. Результати обчислень за вимогами користувача на один день

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Обчислення трудомісткості розробки програмного забезпечення

Вхідні дані [22-23]:

q – передбачуване число операторів – 1700.

C – коефіцієнт складності програми – 1,5.

p – коефіцієнт корекції програми в ході її розробки – 0,08.

B – коефіцієнт збільшення витрат – 1,3.

k – коефіцієнт кваліфікації програміста – 0,8.

$С_{ПР}$– середня годинна заробітна плата програміста, грн/год. Згідно зі статистичними даними, для програміста кваліфікації Junior Software Engineer на C#/.NET [24] – в середньому 800$ на місяць. При 40 годинному робочому тижні та чинному курсі Національного банку України 1 долар = 29,46 гривень, отримаємо значення 163 грн/год.

B_k– число виконавців – 1.

F_p– місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$ годин).

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год. З урахуванням амортизації складових частин ЕОМ та при чинному тарифі на електроенергію для населення [25], при споживанні до 250 КВт – 1,44 грн за КВт/год – 4,5 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою (3.1):

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин}, \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50),

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{\text{отл}}$ – витрати праці на налагодження програми на ЕОМ,

$t_{\text{д}}$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється. Умовне число операторів (підпрограм) визначається за формулою (3.2):

$$Q = q * C * (1 + p), \tag{3.2}$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1700 * 1{,}5 * (1 + 0{,}08) = 2754$$

Витрати праці на вивчення опису задачі t_u визначається за формулою (3.3) з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{80 * k}, \text{людино} - \text{годин}, \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{2754 * 1{,}3}{80 * 0{,}8} = 55{,}94, \text{людино} - \text{годин}.$$

Витрати праці на розробку алгоритму рішення задачі обчислюються за формулою (3.4):

$$t_a = \frac{Q}{20 * k}, \text{людино} - \text{годин}. \quad (3.4)$$

$$t_a = \frac{2754}{20 * 0{,}8} = 172{,}13, \text{людино} - \text{годин}.$$

Витрати на складання програми по готовій блок-схемі визначаються за формулою (3.5):

$$t_n = \frac{Q}{23 * k}, \text{людино} - \text{годин}. \quad (3.5)$$

$$t_n = \frac{2754}{23 * 0{,}8} = 149{,}67, \text{людино} - \text{годин}.$$

Витрати праці на налагодження програми на ЕОМ обчислюються за формулою (3.6):

$$t_{\text{отл}} = \frac{Q}{4 * k}, \text{людино} - \text{годин}. \quad (3.6)$$

$$t_{\text{отл}} = \frac{2754}{4 * 0{,}8} = 860{,}63, \text{людино} - \text{годин}.$$

Витрати праці на підготовку документації обчислюються за формулою (3.7):

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \quad (3.7)$$

де $t_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису визначається формулою (3.8).

$$t_{\text{др}} = \frac{Q}{15 * k}, \text{людино} - \text{годин}, \tag{3.8}$$

$$t_{\text{др}} = \frac{2754}{15 * 0{,}8} = 229{,}50, \text{людино} - \text{годин}.$$

$t_{\text{до}}$ – трудомісткість редагування, печатки й оформлення документації визначається формулою (3.9).

$$t_{\text{до}} = 0{,}75 * t_{\text{др}}, \text{людино} - \text{годин}. \tag{3.9}$$

$$t_{\text{до}} = 0{,}75 * 229{,}50 = 172{,}13, \text{людино} - \text{годин}.$$

Звідси витрати праці на підготовку документації:

$$t_{\text{д}} = 229{,}50 + 172{,}13 = 401{,}63, \text{людино} - \text{годин}.$$

Трудомісткість розробки ПЗ, відповідно, дорівнює:

$$t = 50 + 55{,}94 + 172{,}13 + 149{,}67 + 860{,}63 + 401{,}63$$
$$= 1689{,}99, \text{людино} - \text{годин}$$

3.2. Обчислення витрат на створення програмного забезпечення

Витрати на створення ПЗ $\text{К}_{\text{ПО}}$ включають витрати на заробітну плату виконавця програми $\text{З}_{\text{ЗП}}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ та визначаються за формулою (3.10):

$$\text{К}_{\text{ПО}} = \text{З}_{\text{ЗП}} + \text{З}_{\text{МВ}}, \textit{грн.} \tag{3.10}$$

Заробітна плата виконавців визначається за формулою (3.11):

$$\text{З}_{\text{ЗП}} = t * \text{С}_{\text{ПР}}, \text{грн}, \tag{3.11}$$

де: t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1689{,}99 * 163 = 275\ 468{,}29, \text{грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{мч}, \text{грн,} \qquad (3.12)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год,

$С_{мч}$ - вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 860{,}63 * 4{,}5 = 3\ 872{,}81, \text{грн.}$$

Відповідно, витрати на створення ПЗ становлять:

$$К_{ПО} = 275\ 468{,}29 + 3\ 872{,}81 = 279\ 341{,}11, \text{грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{міс,} \qquad (3.13)$$

де B_k - число виконавців,

F_p - місячний фонд робочого часу.

$$T = \frac{1689{,}99}{1 * 176} = 9{,}6, \text{міс.}$$

Висновки: трудомісткість розробленого програмного додатку становить 1689,99 людино-години. Обчислена вартість роботи по створенню програми

дорівнює 279 341,11 гривень. Визначений затрачений час на створення програмного забезпечення становить 9,6 місяців.

ВИСНОВКИ

Метою кваліфікаційної роботи є створення програмного забезпечення призначеного спростити та пришвидшити процес знаходження оптимальних значень кількісного та якісного складу продуктового кошику виходячи з наявних обмежень.

База даних, призначена для зберігання та маніпулювання даними, представлена у вигляді іменованої сукупності даних, організованих за певними правилами, та містить загальні правила опису, зберігання та маніпулювання даними.

Програма призначена для використання в побуті людьми, які мають бажання чи яким необхідно контролювати рівень цукру в крові і надає можливість виконувати наступні дії:

– проводити оптимізацію свого раціону харчування;

– створювати обліковий запис у системі для збереження результатів оптимізації;

– авторизуватися в системі для отримання можливості додавати користувальницькі продукти харчування та обмеження щодо їхнього споживання;

– додавати та редагувати користувальницькі продукти харчування та обмеження щодо їхнього споживання;

– редагувати свої персональні дані;

– переглядати список стандартних та користувальницьких продуктів та обмежень;

– переглядати список наявних користувачів системи;

– видаляти облікові записи користувачів;

– додавати нові продукти до списку стандартних продуктів харчування;

– додавати нові стандартні обмеження щодо споживання продуктів харчування.

Актуальність поставленої задачі обумовлюється необхідністю людини слідкувати за станом свого здоров’я, складністю виконання ручних обчислень оптимального раціону та відсутності у вільному доступі комплексного програмного забезпечення для виконання такої задачі.

Структура програми являє собою клієнтський додаток, написаний мовою програмування високого рівня C#, що взаємодіє з базою даних «dbDiabet» за допомогою технології Entity Framework Core. База даних розроблена мовою SQL в системі управління базами даних PostgreSQL.

В «Економічному розділі» обчислено трудомісткість розробленого програмного додатку (1689,99 людино-годин), та вартість роботи по створенню програми (279 341,11 гривень). Визначено, що затрачений час на створення програмного додатку становить 9,6 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фітнес Україна – Fitness UA. / URL: https://fitness.org.ua/hlibna-odinica-cho-take-tablicy-i-normy/ (дата звернення 23.05.2022).
2. Хлібні одиниці при цукровому діабеті: таблиця розрахунку, меню дієти для діабетиків. EuroMD / URL: https://euromd.com.ua/hlibni-odinici-pri-cykrovomy-diabeti-tablicia-rozrahynky-menu-dieti-dlia-diabetikiv/ (дата звернення 23.05.2022).
3. Следим за углеводами в пище: что такое хлебные единицы. Relax / URL: https://relax.com.ua/what-to-cook/cooking-advices/sledim-za-uglevodami-v-pishhe-chto-takoe-hlebnye-edinitsy/ (дата звернення 23.05.2022).
4. Как рассчитывать хлебные единицы. LikarInfo / URL: https://www.likar.info/endokrinologiya/article-62021-kak-rasschityvat-hlebnye-edinitsy/ (дата звернення 23.05.2022).
5. Optimization Toolbox 2.2 Руководство пользователя / URL: http://web.archive.org/web/20160811155650/http://www.matlab.exponenta.ru/optimiz/book_1/15.php (дата звернення 24.05.2022).
6. Паттерн MVVM. Metaint.com / URL: https://metanit.com/sharp/wpf/22.1.php (дата звернення 24.05.2022).
7. Model-View-ViewModel (MVVM) Explained. Atmosera / URL: https://www.atmosera.com/blog/model-view-viewmodel-mvvm-explained/ (дата звернення 24.05.2022).
8. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание = Pro WPF 4.5 in C# 2012: Windows Presentation Foundation in .NET 4.5, 4th edition. — М.: «Вильямс», 2013. — 1024 с.
9. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 3.5 с примерами на C# 2008 для профессионалов = Pro WPF in C# 2008: Windows Presentation Foundation with .NET 3.5. — 2-ое. — М.: «Вильямс», 2008. — С. 25. — 928 с.

10. Введение в WPF. Metaint.com / URL: https://metanit.com/sharp/wpf/1.php (дата звернення 26.05.2022).

11. Основні принципи ООП. / URL : https://training.epam.ua/#!/News/275?lang=ua (дата звернення 24.05.2022).

12. Краткий обзор языка C#. Microsoft | Docs / URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата звернення 25.05.2022).

13. Command-line interface (CLI). / URL: https://www.techtarget.com/searchwindowsserver/definition/command-line-interface-CLI (дата звернення 25.05.2022).

14. Common Language Runtime (CLR) overview. Microsoft | Docs / URL: https://docs.microsoft.com/en-us/dotnet/standard/clr (дата звернення 25.05.2022).

15. Visual Studio. Microsoft | Visual Studio / URL: https://visualstudio.microsoft.com/ru/ (дата звернення 25.05.2022).

16. Документация по Entity Framework. Microsoft | Docs / URL: https://docs.microsoft.com/ru-ru/ef/ (дата звернення 25.05.2022).

17. Language Integrated Query (LINQ) (C#). Microsoft | Docs / URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ (дата звернення 25.05.2022).

18. Васильев А. Ю. Работа с PostgreSQL: настройка и масштабирование. Creative Commons Attribution-Noncommercial 4.0 International, 2017. 288 с.

19. SQL:2008 now an approved ISO International Standard | SYBASE URL : https://web.archive.org/web/20110628130925/http://iablog.sybase.com/paulley/2008/07/sql2008-now-an-approved-iso-international-standard/ (дата звернення 26.05.2022).

20. Jim Gray The Transaction Concept: Virtues and Limitations. Tandem Computers Incorporated, 1981. 25 с.

21. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1 | Хабр / URL: https://habr.com/ru/post/282764/ (дата звернення 26.05.2022).

22. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 121 Інженерія програмного забезпечення / О.С. Шевцова, І.М. Удовик; Д : НТУ «Дніпровська політехніка», 2021. – 65 с.

23. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Уклад. О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 11 с.

24. Статистика зарплат програмістів, тестувальників і PM в Україні | DOU / URL: https://jobs.dou.ua/salaries/?period=2021-12&position=Junior%20SE&technology=C#/.NET&experience=0-2 (дата звернення 06.06.2022).

25. Тарифы на электроэнергию в 2022 году. / URL: https://index.minfin.com.ua/tariff/electric/ (дата звернення 06.06.2022).

ДОДАТОК А

КОД ПРОГРАМИ

Сервіс для взаємодії з даними про продукт (товар) в магазині

```
using System;
using System.Collections.Generic;
using System.Linq;
using WpfApp2TypeDiabet.DBServices;
using WpfApp2TypeDiabet.Models;
namespace WpfApp2TypeDiabet.Services
{
   public class GoodInShopService
   {
      //властивість для отримання та встановлення значеннь про товар в магазині
      public GoodInShop { get; set; }
      //властивість для отримання та встановлення значень про список товарів в магазині
      public List<GoodInShop> Goods { get; set; }
      //метод для створення товару в магазині
      public string CreateGoodInShop(User, int goodID, double goodPrice, double goodAmount, string?
goodUnits, int restrictionID)
      {
         try
         {
            if (user.IsSuperUser)
            {
               GoodInShop = new GoodInShop()
               {
                  GoodId = goodID,
                  GoodPrice = goodPrice,
                  GoodAmount = goodAmount,
                  GoodUnits = goodUnits,
                  IsDefault = true,
                  RestrictionID = restrictionID
               };
            }
            else
            {
               GoodInShop = new GoodInShop()
               {
                  GoodId = goodID,
                  GoodPrice = goodPrice,
                  GoodAmount = goodAmount,
                  GoodUnits = goodUnits,
                  IsDefault = false,
                  RestrictionID = restrictionID
               };
            }
            using (ApplicationContext db = new ApplicationContext())
            {
               var goodInShop = from g in db.GoodInShop
                                orderby g.id
                                select g;
               GoodInShop.id = goodInShop.Last().id + 1;
```

```
            db.GoodInShop.Add(GoodInShop);
            db.SaveChanges();
        }
        return "Success";
    }
    catch(Exception e)
    {
        return e.Message;
    }
}
//метод для видалення товару з магазину
public string DeleteGoodInShop()
{
    try
    {
        using (ApplicationContext db = new ApplicationContext())
        {
            foreach(GoodInShop goodInShop in Goods)
            {
                db.GoodInShop.Remove(goodInShop);
                db.SaveChanges();
            }
        }
        return "Success";
    }
    catch (Exception e)
    {
        return e.Message;
    }
}
//метод для отримання значень про товар в магазині за об'єктом товаром
public void GetGoodInShop(Goods good)
{
    Goods = new List<GoodInShop>();
    using (ApplicationContext db = new ApplicationContext())
    {
        var goodInShop = from b in db.GoodInShop
                         where b.GoodId == good.id
                         select b;
        foreach(GoodInShop product in goodInShop)
        {
            Goods.Add(product);
        }
    }
}
//метод для отримання даних про товар в магазині за його ідентифіктатором
public GoodInShop GetGoodInShop(int ID)
{
    GoodInShop = new GoodInShop();
    using (ApplicationContext db = new ApplicationContext())
    {
        var goodInShop = from b in db.GoodInShop
                         where b.id == ID
                         select b;
        GoodInShop = goodInShop.First();
    }
    return GoodInShop;
```

```
        }
        //метод для отримання даних про товар за  ідентифікатором товару в магазині
        public Goods GetGoodByShopID(int goodInShopID)
        {
            Goods targetGood;
            using (ApplicationContext db = new ApplicationContext())
            {
                var goods = from b in db.GoodInShop
                            join good in db.Goods on b.GoodId equals good.id
                                where b.id == goodInShopID
                            select good;
                targetGood = goods.First();
            }
            return targetGood;
        }
        //метод для отримання кількості хлібних одиниць товару з ідентифікатором товару в магазині
        public double GetGoodBUByShopID(int goodInShopID)
        {
            double BU;
            using (ApplicationContext db = new ApplicationContext())
            {
                var BUs = from b in db.GoodInShop
                            join goodShopState in db.GoodShopState on b.id equals goodShopState.GoodInShopID
                            join goodState in db.GoodState on goodShopState.GoodStateID equals goodState.id
                            where b.id == goodInShopID
                            select goodState.Carbohydrates;
                BU = BUs.First();
            }
            return BU;
        }
    }
}
```

Сервіс для взаємодії з даними про користувача

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using WpfApp2TypeDiabet.DBServices;
using WpfApp2TypeDiabet.Models;

namespace WpfApp2TypeDiabet.Services
{
    public class UserService
    {
        //властивість для отримання та встановлення значень поточного користувача
        public User CurrentUser { get; set; }
        //метод перевірки доступності імені в базі даних при створенні нового коритсувача
        public bool IsUserNameAvaliable(User newUser)
        {
            using (ApplicationContext db = new ApplicationContext())
            {
                var users = from b in db.Users
                            where b.UserName.Equals(newUser.UserName)
                            select b;
```

```
            if (users.Any())
            {
                return false;
            }
            else
            {
                return true;
            }
        }
    }
    //метод створення об'єкту класу "користувач"
    public void CreateUser(User newUser)
    {
        using (ApplicationContext db = new ApplicationContext())
        {
            db.Users.Add(newUser);
            db.SaveChanges();
        }
    }
    //метод обробки спроби авторизації
    public User AttemptToLogin(string userName, string password)
    {
        using (ApplicationContext db = new ApplicationContext())
        {
            var user = from b in db.Users
                       where b.UserName.Equals(userName) && b.Password.Equals(password)
                       select b;
            if(user.Any())
            {
                return user.First();
            }
            else
            {
                return null;
            }
        }
    }
    //метод для виходу з облікового запису користувача
    public void LogOut()
    {
        CurrentUser = null;
    }
    //метод для оновлення даних про користувача в базі даних
    public string UpdateDBInfo(string username, string password, int age, double height, double
weight, string gender)
    {
        using (ApplicationContext db = new ApplicationContext())
        {
            var user = from b in db.Users
                       where b.id == CurrentUser.id
                       select b;
            if (!user.First().UserName.Equals(username))
            {
                user.First().UserName = username;
            }
            if (!user.First().Password.Equals(password))
            {
```

```
            user.First().Password = password;
        }
        if (user.First().Age != age)
        {
            user.First().Age = age;
        }
        if (user.First().Height != height)
        {
            user.First().Height = height;
        }
        if (user.First().Weight != weight)
        {
            user.First().Weight = weight;
        }
        if (!user.First().Gender.Equals(gender))
        {
            user.First().Gender = gender;
        }
        try
        {
            db.SaveChanges();
            CurrentUser = GetUser(CurrentUser.id);
            return "Success";
        }
        catch(Exception e)
        {
            return e.Message;
        }
    }
}
//метод для оновлення даних про користувача
public void UpdateInfo(int age, double height, double weight, string gender)
{
    CurrentUser.Age = age;
    CurrentUser.Height = height;
    CurrentUser.Weight = weight;
    CurrentUser.Gender = gender;
}
////метод для отримання об'єкту класу "користувач" за ідентифікатором користувача
public User GetUser(int id)
{
    using (ApplicationContext db = new ApplicationContext())
    {
        var user = from b in db.Users
                   where b.id == id
                   select b;
        if (user.Any())
        {
            return user.First();
        }
        else
        {
            return null;
        }
    }
}
//метод для видалення об'єкту класу "користувач" з бази даних
```

```
        public string DeleteUser(User userToDelete)
        {
            using (ApplicationContext db = new ApplicationContext())
            {
                try
                {
                    db.Users.Remove(userToDelete);
                    db.SaveChanges();
                    return "Success";
                }
                catch (Exception e)
                {
                    return e.Message;
                }
            }
        }
        //метод для отримання списку користувачів
        public ObservableCollection<User> GetUsersList()
        {
            ObservableCollection<User> users = new ObservableCollection<User>();
            using (ApplicationContext db = new ApplicationContext())
            {
                var DbUsers = from b in db.Users
                              select b;
                foreach(User user in DbUsers)
                {
                    users.Add(user);
                }
            }
            return users;
        }
        //метод для отримання об'єкту класу "користувач" за іменем користувача
        public User GetUser(string Name)
        {
            using (ApplicationContext db = new ApplicationContext())
            {
                var user = from b in db.Users
                           where b.UserName == Name
                           select b;
                if (user.Any())
                {
                    return user.First();
                }
                else
                {
                    return null;
                }
            }
        }
    }
}
```

Сервіс для навігації

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Windows.Controls;

namespace WpfApp2TypeDiabet.Services
{
   public class NavigationService
   {
      //подія зміни сторінки
      public event Action<Page> OnPageChanged;
      //стек для збереження історії відвідувань
      private Stack<Page> navigationHistory;
      //метод для перевірки можливості повернутися на попередню сторінку
      public bool CanGoBack => navigationHistory.Any();
      //конструктор класу
      public NavigationService()
      {
         navigationHistory = new Stack<Page>();
      }
      //метод навігації на певну сторінку
      public void Navigate(Page page)
      {
         OnPageChanged?.Invoke(page);
         navigationHistory.Push(page);
      }
      //метод повернення на попередню сторінку в історії відвідувань
      public void GoBack()
      {
         navigationHistory.Pop();
         var page = navigationHistory.Peek();
         OnPageChanged?.Invoke(page);
      }
   }
}
```

Контроль введення даних про ім’я користувача

```
using System.Globalization;
using System.Text.RegularExpressions;
using System.Windows.Controls;
namespace WpfApp2TypeDiabet.ValidationRules
{
   public class UserNameValidationRule : ValidationRule
   {
      public override ValidationResult Validate(object value, CultureInfo cultureInfo)
      {
         string input = value as string;
         string ErrorMessage;
         var hasUpperChar = new Regex(@"[A-Z]+");
         var hasLowerChar = new Regex(@"[a-z]+");
         var hasMiniMaxChars = new Regex(@".{4,20}");
         if (string.IsNullOrWhiteSpace(input) || string.IsNullOrEmpty(input))
         {
            ErrorMessage = "Ім'я користувача не може бути порожнім";
            return new ValidationResult(false, ErrorMessage);
         }
         else if (!hasUpperChar.IsMatch(input))
         {
            ErrorMessage = "Ім'я користувача повинно мати хоча б одну велику літеру";
```

```
            return new ValidationResult(false, ErrorMessage);
        }
        else if(!hasLowerChar.IsMatch(input))
        {
            ErrorMessage = "Ім'я користувача повинно мати хоча б одну малу літеру";
            return new ValidationResult(false, ErrorMessage);
        }
        else if(!hasMiniMaxChars.IsMatch(input))
        {
            ErrorMessage = "Ім'я користувача не можу бути коротше 4 та довше 20 символів";
            return new ValidationResult(false, ErrorMessage);
        }
        else
        {
            return new ValidationResult(true, null);
        }

    }
  }
}
```

Контроль введення даних паролю користувача

```
using System;
using System.Globalization;
using System.Text.RegularExpressions;
using System.Windows.Controls;
namespace WpfApp2TypeDiabet.ValidationRules
{
  internal class PasswordValidationRule : ValidationRule
  {
    public override ValidationResult Validate(object value, CultureInfo cultureInfo)
    {
      string input = value as string;
      string ErrorMessage;
      if (string.IsNullOrWhiteSpace(input) || string.IsNullOrEmpty(input))
      {
        ErrorMessage = "Пароль не може бути порожнім";
        return new ValidationResult(false, ErrorMessage);
      }

      var hasNumber = new Regex(@"[0-9]+");
      var hasUpperChar = new Regex(@"[A-Z]+");
      var hasMiniMaxChars = new Regex(@".{8,15}");
      var hasLowerChar = new Regex(@"[a-z]+");
      var hasSymbols = new Regex(@"[!@#$%^&*()_+=\[{\]};:<>|./?,-]");

      if (!hasLowerChar.IsMatch(input))
      {
        ErrorMessage = "Пароль повинен містити хоча б одну малу літеру";
        return new ValidationResult(false, ErrorMessage);
      }
      else if (!hasUpperChar.IsMatch(input))
      {
        ErrorMessage = "Пароль повинен містити хоча б одну велику літеру";
        return new ValidationResult(false, ErrorMessage);
      }
```

```
            else if (!hasMiniMaxChars.IsMatch(input))
            {
                ErrorMessage = "Пароль не може бути менше 8 та більше 15 символів";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (!hasNumber.IsMatch(input))
            {
                ErrorMessage = "Пароль повинен містити хоча б одне числове значення (0-9)";
                return new ValidationResult(false, ErrorMessage);
            }

            else if (!hasSymbols.IsMatch(input))
            {
                ErrorMessage = "Пароль повинен містити хоча б один спеціальний символ (!@#$%^&*()_+=[{]};:<>|./?,-)";
                return new ValidationResult(false, ErrorMessage);
            }
            else
            {
                return new ValidationResult(true, null);
            }
            throw new NotImplementedException();
        }
    }
}
```

Контроль введення даних про кількість товару в обмеженні щодо його споживання

```
using System.Globalization;
using System.Text.RegularExpressions;
using System.Windows.Controls;

namespace WpfApp2TypeDiabet.ValidationRules
{
    //клас-нащадок класу ValidationRule
    internal class HeightValidationRule : ValidationRule
    {
        //перевизначений метод валідації даних
        public override ValidationResult Validate(object value, CultureInfo cultureInfo)
        {
            string input = value as string;
            string ErrorMessage;
            var hasNumber = new Regex(@"[0-9]+");

            if (string.IsNullOrEmpty(input) || string.IsNullOrWhiteSpace(input))
            {
                ErrorMessage = "Поле зрісту не може бути порожнім";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (!hasNumber.IsMatch(input))
            {
                ErrorMessage = "Зріст має бути числовим значенням";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (!(double.Parse(input, CultureInfo.InvariantCulture) > 0))
```

```
            {
                ErrorMessage = "Зріст не може дорівнювати нулю";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (double.Parse(input, CultureInfo.InvariantCulture) > 5)
            {
                ErrorMessage = "Завелике значення зрісту";
                return new ValidationResult(false, ErrorMessage);
            }
            else
            {
                return new ValidationResult(true, null);
            }
        }
    }
}
```

Контроль введення даних про зріст користувача

```
using System.Globalization;
using System.Text.RegularExpressions;
using System.Windows.Controls;
namespace WpfApp2TypeDiabet.ValidationRules
{
    //клас-нащадок класу ValidationRule
    internal class HeightValidationRule : ValidationRule
    {
        //перевизначений метод валідації даних
        public override ValidationResult Validate(object value, CultureInfo cultureInfo)
        {
            string input = value as string;
            string ErrorMessage;
            var hasNumber = new Regex(@"[0-9]+");
            if (string.IsNullOrEmpty(input) || string.IsNullOrWhiteSpace(input))
            {
                ErrorMessage = "Поле зрісту не може бути порожнім";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (!hasNumber.IsMatch(input))
            {
                ErrorMessage = "Зріст має бути числовим значенням";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (!(double.Parse(input, CultureInfo.InvariantCulture) > 0))
            {
                ErrorMessage = "Зріст не може дорівнювати нулю";
                return new ValidationResult(false, ErrorMessage);
            }
            else if (double.Parse(input, CultureInfo.InvariantCulture) > 5)
            {
                ErrorMessage = "Завелике значення зрісту";
                return new ValidationResult(false, ErrorMessage);
            }
            else
            {
                return new ValidationResult(true, null);
            }
```

```
        }
      }
    }
```

Форма додавання даних про товар адміністратором

```
    <Page
      x:Class="WpfApp2TypeDiabet.Pages.GoodAddAdminPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:validationrules="clr-namespace:WpfApp2TypeDiabet.ValidationRules"
      Title="GoodAdminPage"
      MinWidth="400"
      MinHeight="450"
      DataContext="{Binding Source={StaticResource VMLocator},
Path=_goodAdminPageViewModel}"
      mc:Ignorable="d">
      <Grid Margin="5,0,5,0">
        <Grid.RowDefinitions>
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
          <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
        <Menu Grid.Row="0" Height="Auto">
          <MenuItem Header="Навігація">
            <MenuItem Command="{Binding GoBackCommand}" Header="На попередню">
              <MenuItem.Icon>
                <Image Source="../Images/GoBack.png" Stretch="UniformToFill" />
              </MenuItem.Icon>
            </MenuItem>
            <MenuItem Command="{Binding GoToMainPageCommand}" Header="На головну">
              <MenuItem.Icon>
                <Image Source="../Images/Home.png" Stretch="UniformToFill" />
              </MenuItem.Icon>
            </MenuItem>
          </MenuItem>
        </Menu>
        <TextBlock
          Grid.Row="1"
          HorizontalAlignment="center"
          FontSize="24"
          Text="Додавання даних про новий продукт" />
        <Grid Grid.Row="2" Margin="0,25,0,0">
          <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
          </Grid.RowDefinitions>
```

```
            <TextBlock
                Grid.Row="0"
                FontSize="14"
                Text="Назва товару" />
            <TextBox
                x:Name="GoodNameTextBox"
                Grid.Row="1"
                Height="22"
                Margin="0,5,0,0"
                VerticalAlignment="Top"
                Style="{StaticResource GeneralTextBoxErrorStyle}">
                <TextBox.Text>
                    <Binding Path="GoodName" UpdateSourceTrigger="PropertyChanged">
                        <Binding.ValidationRules>
                            <validationrules:GoodNameValidationRule />
                        </Binding.ValidationRules>
                    </Binding>
                </TextBox.Text>
            </TextBox>
        </Grid>
        <Grid Grid.Row="3" Margin="0,10,0,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
            </Grid.RowDefinitions>
            <TextBlock
                Grid.Row="0"
                FontSize="14"
                Text="Категорія товару" />
            <ComboBox
                x:Name="GoodCategoryComboBox"
                Grid.Row="1"
                Margin="0,5,0,0"
                ItemsSource="{Binding GoodCategoryList}"
                SelectedIndex="0"
                SelectedValue="{Binding GoodCategory, UpdateSourceTrigger=PropertyChanged,
ValidatesOnDataErrors=True}"
                Style="{StaticResource GeneralComboBoxErrorStyle}" />
        </Grid>
        <Grid Grid.Row="4" Margin="0,10,0,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
            </Grid.RowDefinitions>
            <TextBlock
                Grid.Row="0"
                FontSize="14"
                Text="Стан товару" />
            <ComboBox
                x:Name="GoodStateComboBox"
                Grid.Row="1"
                Margin="0,5,0,0"
                ItemsSource="{Binding GoodStateList}"
                SelectedIndex="0"
                SelectedItem="{Binding GoodState, UpdateSourceTrigger=PropertyChanged,
ValidatesOnDataErrors=True}"
                Style="{StaticResource GeneralComboBoxErrorStyle}" />
```

```
</Grid>
<Grid Grid.Row="5" Margin="0,10,0,0">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
  </Grid.RowDefinitions>
  <TextBlock
    Grid.Row="0"
    FontSize="14"
    Text="Ціна одиниці товару, грн" />
  <TextBox
    x:Name="GoodPriceTextBox"
    Grid.Row="1"
    Margin="0,5,0,0"
    Style="{StaticResource GeneralTextBoxErrorStyle}">
    <TextBox.Text>
      <Binding Path="GoodPrice" UpdateSourceTrigger="PropertyChanged">
        <Binding.ValidationRules>
          <validationrules:PriceValidationRule />
        </Binding.ValidationRules>
      </Binding>
    </TextBox.Text>
  </TextBox>
</Grid>
<Grid Grid.Row="6" Margin="0,10,0,0">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
  </Grid.RowDefinitions>
  <TextBlock
    Grid.Row="0"
    FontSize="14"
    Text="Кількість товару" />
  <TextBox
    x:Name="GoodAmountTextBox"
    Grid.Row="1"
    Margin="0,5,0,0"
    Style="{StaticResource GeneralTextBoxErrorStyle}">
    <TextBox.Text>
      <Binding Path="GoodAmount" UpdateSourceTrigger="PropertyChanged">
        <Binding.ValidationRules>
          <validationrules:AmountValidationRule />
        </Binding.ValidationRules>
      </Binding>
    </TextBox.Text>
  </TextBox>
</Grid>
<Grid Grid.Row="7" Margin="0,10,0,0">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
  </Grid.RowDefinitions>
  <TextBlock
    Grid.Row="0"
    FontSize="14"
    Text="Одиниці виміру товару" />
  <ComboBox
```

```
                x:Name="GoodUnitsComboBox"
                Grid.Row="1"
                Margin="0,5,0,0"
                ItemsSource="{Binding GoodUnitList}"
                SelectedIndex="0"
                SelectedItem="{Binding GoodUnits, UpdateSourceTrigger=PropertyChanged,
ValidatesOnDataErrors=True}"
                Style="{StaticResource GeneralComboBoxErrorStyle}" />
            </Grid>
            <Grid Grid.Row="8" Margin="0,10,0,0">
              <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" SharedSizeGroup="InputFields" />
              </Grid.RowDefinitions>
              <TextBlock
                Grid.Row="0"
                FontSize="14"
                Text="Кількість вуглеводів у 100г товару" />
              <TextBox
                x:Name="GoodCarbohydratesTextBox"
                Grid.Row="1"
                Margin="0,5,0,0"
                Style="{StaticResource GeneralTextBoxErrorStyle}">
                <TextBox.Text>
                  <Binding Path="GoodCarbohydrates" UpdateSourceTrigger="PropertyChanged">
                    <Binding.ValidationRules>
                      <validationrules:CarbohydratesValidationRule />
                    </Binding.ValidationRules>
                  </Binding>
                </TextBox.Text>
              </TextBox>
            </Grid>
            <CheckBox
              Grid.Row="9"
              Margin="0,10,0,0"
              Content="Стандартний продукт"
              IsChecked="True"
              IsEnabled="False" />
            <StackPanel
              Grid.Row="10"
              Margin="0,25,0,10"
              HorizontalAlignment="Center"
              Orientation="Horizontal">
              <Button Command="{Binding ConfirmGoodAddAdminCommand}"
Content="Підтвердити">
                <Button.Style>
                  <Style TargetType="Button">
                    <Setter Property="IsEnabled" Value="False" />
                    <Style.Triggers>
                      <MultiDataTrigger>
                        <MultiDataTrigger.Conditions>
                          <Condition Binding="{Binding Path=(Validation.HasError),
ElementName=GoodNameTextBox}" Value="False" />
                          <Condition Binding="{Binding Path=(Validation.HasError),
ElementName=GoodCategoryComboBox}" Value="False" />
                          <Condition Binding="{Binding Path=(Validation.HasError),
ElementName=GoodStateComboBox}" Value="False" />
```

```
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=GoodPriceTextBox}" Value="False" />
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=GoodAmountTextBox}" Value="False" />
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=GoodUnitsComboBox}" Value="False" />
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=GoodCarbohydratesTextBox}" Value="False" />
                      </MultiDataTrigger.Conditions>
                      <Setter Property="IsEnabled" Value="True" />
                    </MultiDataTrigger>
                  </Style.Triggers>
                </Style>
              </Button.Style>
            </Button>
            <Button
              Margin="5,0,0,0"
              Command="{Binding CancelGoodAddAdminCommand}"
              Content="Скасувати" />
          </StackPanel>
        </Grid>
      </Page>
```

Модель представлення форми авторизації

```
using DevExpress.Mvvm;
using System.Windows;
using System.Windows.Input;
using WpfApp2TypeDiabet.Models;
using WpfApp2TypeDiabet.Pages;
using WpfApp2TypeDiabet.Services;

namespace WpfApp2TypeDiabet.ViewModels
{
  public class LoginPageViewModel : BindableBase
  {
    private readonly NavigationService _navigation;
    private readonly UserService _userService;
    //властивість для отримання та встановлення значень логіну користувача
    public string Login {get; set;}
    //властивість для отримання та встановлення значень паролю користувача
    public string Password {get; set;}
    //конструктор класу
    public LoginPageViewModel(NavigationService navigation, UserService userSevice)
    {
      _navigation = navigation;
      _userService = userSevice;
    }
    //команда переходу на форму авторизації
    public ICommand GoToRegistrationPageCommand => new DelegateCommand(() =>
    {
      if (!string.IsNullOrEmpty(Login) || !string.IsNullOrEmpty(Password))
      {
        MessageBoxResult result = MessageBox.Show("Ви впевнені, що хочете скасувати авторизацію?",
        "Скасування авторизації", MessageBoxButton.YesNo, MessageBoxImage.Question);
        if (result == MessageBoxResult.Yes)
```

```
                {
                    ClearFields();
                    _navigation.Navigate(new RegistrationPage());
                }
            }
            else
            {
                ClearFields();
                _navigation.Navigate(new RegistrationPage());
            }
        });
        //команда переходу на форму запиту на авторизацію
        public ICommand GoToPromptPageCommand => new DelegateCommand(() =>
        {
            if (!string.IsNullOrEmpty(Login) || !string.IsNullOrEmpty(Password))
            {
                MessageBoxResult result = MessageBox.Show("Ви впевнені, що хочете скасувати авторизацію?",
                "Скасування авторизації", MessageBoxButton.YesNo, MessageBoxImage.Question);
                if (result == MessageBoxResult.Yes)
                {
                    ClearFields();
                    _navigation.Navigate(new PromptToLogin());
                }
            }
            else
            {
                ClearFields();
                _navigation.Navigate(new PromptToLogin());
            }
        });
        //команда авторизації
        public ICommand LoginCommand => new DelegateCommand(() =>
        {
            User userToLogin = _userService.AttemptToLogin(Login, Password);
            if (userToLogin != null)
            {
                _userService.CurrentUser = userToLogin;
                if (userToLogin.IsSuperUser)
                {
                    _navigation.Navigate(new AdminMainPage());
                }
                else
                {
                    _navigation.Navigate(new UserMainPage());
                }
                ClearFields();
            }
            else
            {
                MessageBox.Show("Неможливо авторизуватися!", "Помилка авторизації",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }, ()=>!string.IsNullOrWhiteSpace(Login) && !string.IsNullOrWhiteSpace(Password));
        //метод очищення полів
        private void ClearFields()
        {
```

```
            if(!string.IsNullOrEmpty(Login))
            {
                Login = string.Empty;
            }
            if(!string.IsNullOrEmpty(Password))
            {
                Password = string.Empty;
            }
        }
    }
}
```

Модель користувача

```
using System.Collections.ObjectModel;
namespace WpfApp2TypeDiabet.Models
{
    public class User
    {
        //властивість для встановлення та отримання значень ідентифікатора користувача
        public int id { get; set; }
        //властивість для встановлення та отримання значень імені користувача
        public string UserName { get; set; }
        //властивість для встановлення та отримання значень паролю користувача
        public string Password { get; set; }
        //властивість для встановлення та отримання значень віку користувача
        public int Age { get; set; }
        //властивість для встановлення та отримання значень зрісту користувача
        public double Height { get; set; }
        //властивість для встановлення та отримання значень ваги користувача
        public double Weight { get; set; }
        //властивість для встановлення та отримання значень статі користувача
        public string Gender { get; set; }
        //властивість для встановлення та отримання значень показника, чи є користувач адміністратором
        public bool IsSuperUser { get; set; }
        //конструктор з параметрами класу
        public User(string userName, string password, int age, double height, double weight,
            string gender, bool isSuperUser)
        {
            UserName = userName;
            Password = password;
            Age = age;
            Height = height;
            Weight = weight;
            Gender = gender;
            IsSuperUser = isSuperUser;
        }
        //властивість для встановлення та отримання списку товарі користувача
        public ObservableCollection<UserGoodList> CustomGoods { get; set; } = new ObservableCollection<UserGoodList>();
        //властивість для встановлення та отримання списку обмежень користувача
        public ObservableCollection<UserRestrictionList> CustomRestrictions { get; set; } = new ObservableCollection<UserRestrictionList>();
        //властивість для встановлення та отримання списку продуктових кошиків користувача
        public ObservableCollection<GoodBasket> GoodBaskets { get; set; } = new ObservableCollection<GoodBasket>();
```

```
    }
}
```

Користувальницьких компонент BindablePasswordBox

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
namespace WpfApp2TypeDiabet.Components
{
    /// <summary>
    /// Логика взаимодействия для BindablePasswordBox.xaml
    /// </summary>
    public partial class BindablePasswordBox : UserControl
    {
        private bool _isPasswordChanging;
        public static readonly DependencyProperty PasswordProperty =
            DependencyProperty.Register("Password", typeof(string), typeof(BindablePasswordBox),
                new FrameworkPropertyMetadata(string.Empty,
FrameworkPropertyMetadataOptions.BindsTwoWayByDefault,
                    PasswordPropertyChanged, null, false, UpdateSourceTrigger.PropertyChanged));
        //обробник події зміни властивості компоненту
        private static void PasswordPropertyChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
        {
            if (d is BindablePasswordBox passwordBox)
            {
                passwordBox.UpdatePassword();
            }
        }
        //властивість отримання та встановлення значення паролю
        public string Password
        {
            get { return (string)GetValue(PasswordProperty); }
            set { SetValue(PasswordProperty, value); }
        }
        //конструктор класу
        public BindablePasswordBox()
        {
            InitializeComponent();
        }
        //обробник події зміни пароля в компоненті
        private void PasswordBox_PasswordChanged(object sender, RoutedEventArgs e)
        {
            _isPasswordChanging = true;
            Password = passwordBox.Password;
            _isPasswordChanging = false;
        }
        //метод оновлення паролю
        private void UpdatePassword()
        {
            if (!_isPasswordChanging)
            {
                passwordBox.Password = Password;
            }
        }
    }}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_Коваленко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_Коваленко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Програма_Коваленко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Презентація_Коваленко.ppt	Презентація кваліфікаційної роботи